Naïve ABox abduction in $\mathcal{ALC}$ using a DL tableau

Ken Halland[1][2] and Katarina Britz[2]

[1] School of Computing, University of South Africa, Pretoria, South Africa
[2] Centre for Artificial Intelligence Research: UKZN and CSIR Meraka Institute, South Africa

hallakj@unisa.ac.za and arina.britz@meraka.org.za

Abstract. The formal definition of abduction asks what needs to be added to a knowledge base to enable an observation to be entailed by the knowledge base. An observation which is not entailed by the knowledge base will result in open branches in a complete semantic tableau for the entailment. The statements required to close these branches therefore represent a solution to the abductive problem.
In this paper we describe how this idea can be implemented for ABox abduction in the description logic $\mathcal{ALC}$. We analyse the limitations of our algorithm and propose refinements to improve the quality of results.

1 Introduction

Abduction is a form of non-standard reasoning where explanations are generated for certain observations in the context of some background knowledge. This is as opposed to deduction – the standard form of reasoning where the logical consequences of some knowledge are determined.

A typical use of abduction is in the process of medical diagnosis. Say a patient displays some symptoms. A doctor uses abductive reasoning to generate hypotheses about the possible ailment(s) causing the symptoms. These hypotheses can then be tested by collecting corroborating evidence so that deduction can be used to make a correct diagnosis.

Abductive reasoning is *non-monotonic* in that the solutions derived from some background knowledge and an observation may no longer hold if we add new statements to the background knowledge.

Different forms of abduction have been formally defined in different logics. In their programmatic paper, Elsenbroich *et al* [6] define and describe various forms of abduction in description logics (DLs). A number of researchers have taken up the challenge and developed algorithms for some of these forms of abduction in selected DLs ([4, 5]).

In particular, Klarman *et al* [8] have provided a resolution-based algorithm and a tableau-based algorithm for performing ABox abduction in $\mathcal{ALC}$. The tableau-based algorithm involves translating the knowledge base and the observation for an abductive problem from its DL specification to first-order logic (FOL), then constructing a FOL connection tableau and harvesting abductive solutions from open branches. These solutions are then translated back to a DL notation.

In this paper, we describe a glass box algorithm for doing this directly by means of a DL semantic tableau. We then show that this is a naïve algorithm in that it fails to generate a number of solutions which could be desirable in certain circumstances. We propose strategies to adapt the algorithm to address these deficiencies. We also define the notion of semantic minimality as a means to compare or rank solutions.

A potential advantage of this approach over Klarman's is that it can utilise optimisation techniques used in DL tableaux.

Section 2 highlights the relevant aspects of the syntax and semantics of $\mathcal{ALC}$, Section 3 includes a definition of ABox abduction in $\mathcal{ALC}$ (slightly more general than Klarman *et al* [8]), and Section 4 gives a description of the standard semantic tableau algorithm for $\mathcal{ALC}$. Section 5 then describes how this algorithm can be adapted to perform ABox abduction. Section 6 explains why the proposed abduction algorithm is naïve and how this can be addressed. Finally, Section 7 discusses the prospects for future work.

2 The Description Logic $\mathcal{ALC}$

Description Logics (*DLs*) are a family of fragments of first-order logic, suitable as knowledge representation formalisms and amenable to the implementation of efficient reasoners [1]. There is a trade-off between the expressivity of different DLs and the efficiency of the algorithms that have been defined to reason over them. $\mathcal{ALC}$ is a DL of medium expressivity.

Syntax: The reader is referred to the Description Logic Handbook [1] for the syntax of $\mathcal{ALC}$. We highlight the following terminology for our current purposes:

A *knowledge base* is a set of *statements* partitioned into an *ABox* and a *TBox*. *ABox assertions* are statements of the form $C(I)$ and $R(I, J)$ (called *concept assertions* and *role assertions*, respectively), and *TBox axioms* are statements of the form $C \sqsubseteq D$ (sometimes called *general concept inclusions*, or *GCIs*).[3]

The following serves as a running example. Admittedly it is not very good knowledge representation, but we have chosen the given formulation for the sake of illustration.

Example 1: The following knowledge base is intended to express the ideas that Influenza A is a form of influenza, Malaria vivax is a form of malaria (caused

[3] In this and the following specifications, C and D represent arbitrary concept descriptions, R represents an arbitrary role name, and I and J represent arbitrary individual names.

by *Plasmodium vivax*), and someone infected with influenza or malaria will be feverish:

$$\mathsf{Influenza}(\mathsf{FLU\_A})$$
$$\mathsf{Malaria}(\mathsf{MAL\_V})$$
$$\exists\mathsf{infectedWith.Influenza} \sqcup \exists\mathsf{infectedWith.Malaria} \sqsubseteq \mathsf{Feverish}$$

Semantics: Once again, the reader is referred to the DL Handbook [1] for the semantics of $\mathcal{ALC}$. We highlight the following terminology for our purposes:

An interpretation $\mathcal{I}$ is a *model* of a knowledge base $\mathcal{K}$ if all the statements of $\mathcal{K}$ are true in $\mathcal{I}$. A concept description C is *satisfiable* with respect to a knowledge base $\mathcal{K}$ if there is some model of $\mathcal{K}$ such that the interpretation of C is not empty. A statement ϕ is *entailed* by a knowledge base $\mathcal{K}$ if ϕ is true in all models of $\mathcal{K}$, in which case we write $\mathcal{K} \models \phi$. In an abuse of notation, we often write $\mathcal{K} \models \Phi$ where Φ is a set of statements. By this we mean that $\mathcal{K} \models \phi$ for all $\phi \in \Phi$. A knowledge base $\mathcal{K}$ is *consistent* if it admits a model.

3 Abduction

An abduction problem is normally defined in terms of an observation (in the form of one or more statements) which is not entailed by a theory (i.e. a set of statements), and asks what needs to be added to the theory to entail the observation.

Example 2: Using the knowledge base given in Example 1, say we observe that John is feverish. An abduction problem would be to ask what should be added to the knowledge base to allow us to infer $\mathsf{Feverish}(\mathsf{JOHN})$.

We expect abduction to allow us to hypothesize that John is infected with influenza or he is infected with malaria, i.e. $\exists\mathsf{infectedWith.Influenza}(\mathsf{JOHN})$ or $\exists\mathsf{infectedWith.Malaria}(\mathsf{JOHN})$. In fact, more specific hypotheses would be that he is infected with Influenza A, i.e. $\mathsf{infectedWith}(\mathsf{JOHN}, \mathsf{FLU\_A})$, or that he is infected with Malaria vivax, i.e. $\mathsf{infectedWith}(\mathsf{JOHN}, \mathsf{MAL\_V})$. The reader might like to check that adding any of these assertions to the knowledge base will allow us to infer $\mathsf{Feverish}(\mathsf{JOHN})$.

One problem with a formal definition of abduction is how to narrow down the possibly infinite number of solutions to an abduction problem. Various criteria have been defined for this purpose. Like other authors [6, 8], we restrict our attention to the following three:

i. *Consistency:* A solution should not create a contradiction with the background knowledge.
ii. *Relevance:* A solution should be expressed in terms of the background knowledge; it shouldn't introduce an independent theory.
iii. *Minimality:* A solution should not hypothesize more than necessary.

Example 3: The solutions given in Example 2 are consistent, relevant and minimal (i.e. minimal at least in a syntactic sense). The following solutions do not comply with these criteria:

i. $\{\neg\mathsf{Malaria}(\mathsf{MAL\_V})\}$. If this assertion were added to the knowledge base, it would cause a contradiction, and would allow us to infer anything. But this would not be a helpful solution.
ii. $\{\exists\mathsf{infectedWith.ScarletFever} \sqsubseteq \mathsf{Feverish}, \exists\mathsf{infectedWith.ScarletFever}(\mathsf{JOHN})\}$. This is an abductive solution, since if both these statements were added to the knowledge base, it would allow us to infer $\mathsf{Feverish}(\mathsf{JOHN})$. However, it would also allow us to make this inference *independently of the knowledge base* and is therefore not a relevant solution. (Incidentally, the observation itself, in this case $\{\mathsf{Feverish}(\mathsf{JOHN})\}$, is also always a non-relevant solution, since adding it to the knowledge base would allow the observation to be trivially inferred.)
iii. $\{\exists\mathsf{infectedWith.Influenza}(\mathsf{JOHN}), \exists\mathsf{infectedWith.Malaria}(\mathsf{JOHN})\}$. Although this is a valid solution, it is not minimal because it hypothesizes too much, namely that John is infected with both influenza and malaria.

3.1 ABox Abduction in $\mathcal{ALC}$

As stated in the introduction, attempts have been made to define abduction and implement reasoners that can make abductive inferences in many logics, including description logics. ABox abduction (as opposed to general or so-called *knowledge base* abduction [6]) asks what ABox assertions need to be added to a DL knowledge base to allow an observation (also in the form of ABox assertions) to be inferred.

The astute reader will note that apart from not being relevant, Example 3 ii is also not an ABox abduction solution, since it contains a TBox axiom.

Definition 1. *Given a knowledge base $\mathcal{K}$ and a set of ABox assertions Φ (both in $\mathcal{ALC}$) such that $\mathcal{K}$ does not entail Φ and $\mathcal{K} \cup \Phi$ is consistent, then a set of ABox assertions Θ is an abductive solution for $(\mathcal{K}, \Phi)$ if $\mathcal{K} \cup \Theta \models \Phi$.*

We can narrow down the solutions in three ways:

i. *Consistency*: $\mathcal{K} \cup \Theta$ is consistent.
ii. *Relevance*: Φ is not entailed by Θ.
iii. *Minimality*: We distinguish two kinds of minimality:
   (a) *Syntactic*: No proper subset of Θ is a solution.
   (b) *Semantic*: There is no non-equivalent solution Θ' such that $\mathcal{K} \cup \Theta \models \mathcal{K} \cup \Theta'$.

Note that our definition of semantic minimality induces a partial ordering on the set of solutions, and that there can be a number of semantically minimal (non-equivalent) solutions to a particular abductive problem. We say that a solution Θ is *closer to semantic minimality* than a solution Θ' if $\mathcal{K} \cup \Theta' \models \mathcal{K} \cup \Theta$ and $\mathcal{K} \cup \Theta \not\models \mathcal{K} \cup \Theta'$.

4 The Semantic Tableau Algorithm for $\mathcal{ALC}$

For a more detailed description of the semantic tableau algorithm for $\mathcal{ALC}$, the reader is referred to the Handbook of Knowledge Representation [2]. We highlight the following terminology for our current purposes:

The standard semantic tableau algorithm for description logics (and for $\mathcal{ALC}$ in particular) tries to find (i.e. construct) a model of the knowledge base by applying so-called *expansion rules* to its statements.

The expansion rules only apply to ABox assertions, so before the algorithm can commence, the TBox axioms in the knowledge base must be converted to concept assertions by a process called *internalisation*.

In $\mathcal{ALC}$, it is possible to specify a knowledge base that has infinite models (by means of a so-called *cyclic* TBox). This issue is dealt with by a technique called *blocking*, which essentially detects when more than one individual has the same labelling in the current model.

If the algorithm detects a *contradiction* (or *clash*), i.e. the current set of assertions contains a concept assertion and its negation, it backtracks and tries another branch of its search. If it gets to a point where the current set of assertions are *saturated*, i.e. no more expansion rules can be applied and there is no contradiction, then a model has been found, the algorithm terminates and reports that the original knowledge base is consistent.

The algorithm described above performs *consistency checking* of a knowledge base. It can easily be adapted to perform the related reasoning task of *instance checking*, i.e. deciding whether a concept assertion is entailed by a knowledge base, as follows: The negation of the concept assertion being tested is added to the knowledge base and the algorithm described above is executed. If the resulting knowledge base is consistent, we conclude that the assertion is not entailed by the knowledge base (and *vice versa*).

5 Adaption of the Semantic Tableau Algorithm for ABox Abduction

For the purpose of ABox abduction, we perform instance checking of an observation by means of a so-called *extended* (or complete) semantic tableau, i.e. a tableau that doesn't terminate when the first open branch is attained. Every time an open branch is attained, the current set of assertions (representing a model of the original set of assertions) is stored, the algorithm backtracks and continues its search. Reiter's minimal hitting set algorithm [10] is then used to generate abductive solutions from these models. Simply put, one unexpanded concept assertion (involving a non-dummy individual) is chosen from each (terminal) open branch. (Dummy individuals are introduced by the $\exists$-expansion rule.) Each combination of the complements of such assertions forms an abductive solution. Finally the algorithm outputs all solutions that are consistent with the knowledge base and that are relevant.

```
Input   : ABox, TBox and Obs
Output: Naïve abduction solutions
A ← negNF(ABox) ∪ negNF(¬Obs);
internalise(TBox, U, A);
M ← {};
SSet ← {};
extendedST(A, U, M);
if M = {} then
    print "The observation follows from the knowledge base";
    return SSet
minimalHS(M, H);
foreach S in H do
    if consistentST(A ∪ S, U) and relevantST(S, Obs) then
        SSet ← SSet ∪ {S}
return SSet
```

Algorithm 1: Naïve ABox abduction algorithm for $\mathcal{ALC}$

Function negNF transforms a set of concept assertions to negation normal form. Procedure internalise takes a set of TBox axioms and transforms them into a set of universal concepts U, each in negation normal form. (Note that in some implementations of the tableau algorithm, all the TBox axioms are converted into one long universal concept. We rather store them as separate concepts – one per TBox axiom – to save having to repeatedly expand the long concept.) The algorithm then applies each of these universal concepts to all the individual names mentioned in the ABox, adding the assertions to A. U is returned via parameter to be used in extendedST whenever a dummy variable is created for the ∃-rule. M is a set of models (where each model is a set of unexpanded assertions obtained from an open branch). Procedure extendedST performs the extended semantic tableau algorithm explained above. Whenever an open branch is attained, it adds the current set of unexpanded assertions to M and backtracks. If the observation is entailed by the knowledge base, then all branches will close and M will be empty. This means that we are not dealing with a proper abduction problem. M is sent to procedure minimalHS to generate the minimal hitting sets and store them in H. minimalHS ensures the syntactic minimality of solutions. Functions consistentST and relevantST are like calls to the semantic tableau procedure (described in Section 4). They determine whether the solution is consistent with the original knowledge base, and whether the observation is not entailed by the solution, respectively.

Although extendedST implements blocking, this is not used in any way for the generation of solutions. The argument is as follows: Since $\mathcal{ALC}$ has the *finite model property* [1], every knowledge base that has an infinite model (handled by blocking) also has at least one finite model (represented by an open branch of the tableau). Since our algorithm closes all open branches and so removes all finite models, the infinite models will also be removed.

5.1 Complexity

The complexity of the standard semantic tableau algorithm for consistency checking with general TBoxes in $\mathcal{ALC}$ is ExpTime [2]. In our extended semantic tableau (in procedure extendedST), the worst case involves maximal branching where every branch is open, since we have to store all the assertion sets of all open branches. Nevertheless, the maximum number of branches is linear in the size of the initial assertion set, and the number of assertions in each such branch is also linear in the size of the initial assertion set. This means that the space required to store all the assertion sets in all the open branches is polynomial in the size of the initial assertion set. This at least means that the space requirements don't blow up to ExpSpace, which means that the extended semantic tableau algorithm is at worst in ExpTime.

Reiter's minimal hitting set algorithm (in general) is NP-complete [11]. In our case (in procedure minimalHS), the number of sets and their size is polynomial in the size of the initial assertion set. This means that the time required in our case is also in NP.

Finally, the algorithm invokes the functions consistentST and relevantST twice for each candidate solution. Although the space required for relevantST is only polynomial (because it does not deal with the TBox), consistentST is in ExpTime in the worst case because it must deal with the TBox. Since the number of hitting sets is polynomial in the size of the initial assertion set, the total space requirement for this process is in ExpTime.

The entire algorithm is therefore in ExpTime.

5.2 Soundness and Completeness

Taking Definition 1 as the standard for ABox abduction in $\mathcal{ALC}$, Algorithm 1 is sound but not complete.

It is sound because all solutions that it generates are proper abduction solutions according to the definition. Consider the following argument: Each solution is a set comprised of the complements of assertions in the open branches of the extended semantic tableau, such that each open branch has a representative in the solution. So if the assertions of such a solution were to be added to the knowledge base (and the satisfiability test were to be performed again), all branches would close, indicating that the observation is now entailed by the knowledge base. This is precisely the definition of an abduction solution.

It is not complete due to the problems detailed in Section 6. One should not be surprised at this because abductive inference is notoriously incomplete due to the often infinite number of solutions to an abduction problem. Narrowing down the spectrum of solutions by means of criteria such as consistency, relevance and minimality only partially addresses this issue. Many solutions within these criteria are difficult to obtain, particularly by means of the techniques described here. In Section 6 we propose workarounds to make the algorithm more complete.

6 Naïvety

The algorithm described in Section 5 is "naïve" in that it doesn't deal with all observations and it doesn't generate all possible solutions allowed by our definition. In particular, the algorithm has the following weaknesses:

1. The observation can only consist of a single concept assertion.
2. Solutions containing disjunctions are not generated.
3. Solutions involving role assertions are not generated.
4. Semantically minimal solutions are not always generated.

We discuss each of these problems in turn:

6.1 Single Concept Assertions

Contrary to Definition 1, our algorithm does not allow more than one concept assertion in the observation. This is because an observation consisting of multiple assertions is really a conjunction of assertions, and the first step of the algorithm is to add the negation of the observation, which is in effect a disjunction of the negations of its individual assertions. We cannot express such a disjunction of assertions in DL syntax when the assertions involve different individuals. (This is not a problem for multiple concept assertions about the same individual, e.g. $C(I)$ and $D(I)$ can be negated as $\neg C \sqcup \neg D(I)$.)

Furthermore, the observation may not contain any role assertions, because $\mathcal{ALC}$ syntax doesn't allow negated role assertions. So we cannot deal with observations such as $\mathsf{hasSymptom}(\mathsf{JOHN}, \mathsf{INTERMITTENT\_FEVER})$. (It is true that in many cases this situation could be handled by alternative modelling, e.g. $\exists\mathsf{hasSymptom}.\mathsf{IntermittentFever}(\mathsf{JOHN})$, but there might be situations where this is not practical or desirable.)

Neither of these situations are a problem for Klarman *et al* [8], since their translation to FOL syntax allows disjunctions of concept assertions as well as negated role assertions.

Proposed Workaround: A brute-force method of dealing with multiple assertions in the observation would be to execute the algorithm once for each such assertion, harvest all the models from all open branches of all executions, and then process them as normal. This, however, would involve a lot of duplication (processing the rest of the assertions repeatedly). One way to avoid such duplication would be to keep the complemented assertions of the observation in a separate list from the other assertions. (The set of complemented assertions would represent a disjunction of assertions, whereas the other set would represent a conjunction of assertions – as normal.) Then when no other expansion rules can be applied to the normal set, the algorithm can branch for one of the complemented assertions.

The negation of a role assertion $R(I, J)$ can be accounted for by two assertions: $\forall R.A(I)$ and $\neg A(J)$, where A is a dummy concept name not occurring in the knowledge base. Adding these two assertions to the initial set of assertions will have the same effect as adding the negation of the role assertion. Assertions involving such dummy concept names will need to be ignored for the purposes of determining abductive solutions.

6.2 Disjunctions

Our algorithm suffers from the same problem as Klarman's [8], namely that the abductive solutions do not contain disjunctions, i.e. assertions of the form $C \sqcup D(I)$. For example, it does not generate the following solution to the problem described in Example 2: $(\exists \mathsf{infectedWith}.\mathsf{Influenza}) \sqcup (\exists \mathsf{infectedWith}.\mathsf{Malaria})(\mathsf{JOHN})$. Note that a solution with such a disjunction is closer to semantic minimality than the corresponding two solutions with the individual disjuncts.

Proposed Workaround: One reason for this problem is that our algorithm only considers unexpanded concept assertions for forming solutions. Allowing expandable concept assertions to be selected for solutions would allow some disjunctions, but not all. For example, say we replaced the axiom of Example 1 with the two axioms $\exists \mathsf{infectedWith}.\mathsf{Influenza} \sqsubseteq \mathsf{Feverish}$ and $\exists \mathsf{infectedWith}.\mathsf{Malaria} \sqsubseteq \mathsf{Feverish}$. In this case, the solution with the disjunction above would be a valid solution, but would not be generated.

One could construct some such solutions from their constituent parts, e.g. $C \sqcup D(I)$ could be constructed from $C(I)$ and $D(I)$, but more complex solutions involving disjunctions inside quantifiers would be more difficult, e.g. $\exists R.(C \sqcup D)(I)$ will not be generated when $\exists R.C(I)$ and $\exists R.D(I)$ are.

Klarman *et al* [8] get around the problem by defining it away. They define ABox abduction in $\mathcal{ALC}$ as only providing solutions in $\mathcal{ALE}$ (a less expressive DL than $\mathcal{ALC}$, i.e. without disjunction and full negation).

6.3 Role Assertions

A more serious weakness is that the algorithm will never generate solutions involving role assertions. So the more specific solutions mentioned in Example 2, namely $\mathsf{infectedWith}(\mathsf{JOHN}, \mathsf{FLU\_A})$ and $\mathsf{infectedWith}(\mathsf{JOHN}, \mathsf{MAL\_V})$, are unattainable with our algorithm.

Stated more generically: Consider a knowledge base containing the ABox assertion $\forall R.A(I)$, and say we want an abductive explanation of the observation $A(J)$. An obvious solution is $R(I, J)$. But our abductive solutions are always the complements of assertions needed to close the open branches of a semantic tableau. Since the tableau algorithm does not infer negated role assertions, this solution will not be generated.

Proposed Workaround: One way would be to add the assertion $R(I, J)$ to a solution whenever the "pattern" $\{\forall R.A(I), \neg A(J)\}$ occurs in a open terminal branch. However, although this will enable some role assertions to be included in solutions, it will not generate all: Adding the role assertion $R(I, J)$ to a knowledge base will cause an open branch containing $\{\forall R.C(I), \neg D(J)\}$, where C is disjoint from D, to close, so it should form part of an abductive solution whenever this pattern occurs. Such a pattern could be difficult to recognise, and the easiest way to deal with this would probably be to allow nominals in the language, since a negated role assertion $\neg R(I, J)$ can then be expressed as a concept assertion, namely $\forall R.\neg\{J\}(I)$ [7].

6.4 Semantic Minimality

This problem is best explained by means of an example. Say we add the axiom $\exists\mathsf{bloodTestIndicates.Plasmodium} \sqsubseteq \exists\mathsf{infectedWith.Malaria}$ to the knowledge base of Example 1.

Using the observation of Example 2, the algorithm now generates the solutions $\exists\mathsf{infectedWith.Influenza(JOHN)}$ and $\exists\mathsf{bloodTestIndicates.Plasmodium(JOHN)}$. One of the solutions we got previously, namely $\exists\mathsf{infectedWith.Malaria(JOHN)}$ has gone! In fact, a solution that is closer to semantic minimality has been lost.

Proposed Workaround: Many such solutions that are closer to semantic minimality can be obtained by allowing expanded concept assertions as part of solutions (including the above example). However, this will not solve all problems: Consider the knowledge base consisting of TBox $= \{A_1 \sqcup A_2 \sqsubseteq A_3, \exists R.A_3 \sqsubseteq A_4\}$ and ABox $= \{R(I, J)\}$, and say we want abductive solutions for the observation $\{A_4(I)\}$. If we apply the algorithm to this problem, three solutions are generated: $\{A_1(J)\}$, $\{A_2(J)\}$ and $\{A_3(J)\}$. If we allow expandable assertions, we get $\{A_1 \sqcup A_2(J)\}$ and $\{\exists R.A_3(I)\}$ as solutions, but not $\{\exists R.A_1(I)\}$ or $\{\exists R.A_2(I)\}$. These are closer to semantic minimality than $\{\exists R.A_3(I)\}$.

Whether we manage to find a way of generating all semantically minimal solutions, or just those attainable by allowing expandable assertions, we imagine that the user of a system implementing an abduction algorithm would want to be able to explore a range of such solutions.

The notion of semantic minimality is related to the notion of weakest sufficient conditions [9], although this work is restricted to propositional logic. It is also reminiscent of work on least common subsumers [3], and we plan to investigate the possibility of applying those ideas to this situation.

7 Future Work

Algorithm 1 does not implement many of the optimizations (e.g. back-jumping and caching) commonly used in DL tableau algorithms. Incorporating these into our algorithm promises to give a real efficiency advantage over the FOL connection tableau used in Klarman's algorithm [8].

This work also promises to be transferable to other more expressive DLs. As stated in Section 6.3, the problem of dealing with role assertions will disappear in languages that allow nominals.

Languages that do not have the finite model property will need some means of dealing with infinite models. (We imagine that the current assertion set at the point of blocking could simply be added to the set of models collected by extendedST so that it will be closed by all solutions.)

As stated in Section 6.4, we also intend to investigate the work on weakest sufficient conditions and least common subsumers for their applicability to ranking solutions.

Acknowledgements

This work was partially funded by a European Union international research staff exchange scheme – Project number 247601, Net2: Network for Enabling Networked Knowledge, from the FP7-PEOPLE-2009-IRSES call. Thanks to Tommie Meyer of the CSIR Meraka Institute (in South Africa) and Enrico Franconi of the Free University of Bozen/Bolzano (in Italy) for infrastructure and support.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook, Cambridge University Press (2003)
2. Baader, F., Horrocks, I. Sattler, U.: Chapter 3: Description Logics. In: van Harmelen, F., Lifschitz, V., Porter, B., editors: Handbook of Knowledge Representation, Elsevier (2007)
3. Baader, F., Sertkaya, B., Turhan, A.: Computing the least common subsumer w.r.t. a background terminology, Journal of Applied Logic, Springer (2004)
4. Di Noia, T., Di Sciascio, E., Donini, F.M.: Computing information minimal match explanations for logic-based matchmaking, in Proc. of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, vol 02, IEEE Computer Society (2009)
5. Du, J., Qi, G., Shen, Y-D., Pan, J.Z.: Towards practical ABox abduction in large OWL DL ontologies, in Proc. of the 25th AAAI Conference (2011)
6. Elsenbroich, C., Kutz, O., Sattler, U.: A case for abductive reasoning over ontologies, in Proc. of the OWLED'06 Workshop, vol 216 (2006)
7. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$, in Proc. of KR2006, pp 57-67 (2006)
8. Klarman, S., Endriss, U., Schlobach, S.: ABox abduction in the description logic $\mathcal{ALC}$, Journal of Automated Reasoning, vol 46:1 (2011)
9. Lin, F.: On strongest necessary and weakest sufficient conditions, in Proc. of KR2000, pp 167-175 (2000)
10. Reiter, R.: A theory of diagnosis from first principles, Artificial Intelligence, vol 32 (1987)
11. Wotawa, F.: A variant of Reiter's hitting-set algorithm, Information Processing Letters, vol 79 (2001)