A Framework for Explaining Reasoning in Description Logics *

Xi Deng and **Volker Haarslev** and **Nematollaah Shiri**
Concordia University
Dept. of Computer Science & Software Engineering
1515 de Maisonneuve West, Montreal, Quebec, Canada, H3G 1M8
{xi_deng,haarslev,shiri}@cse.concordia.ca

Abstract

We present a resolution based framework to explain reasoning in description logics and demonstrate its applicability to explain unsatisfiability and inconsistency queries w.r.t TBoxes and ABoxes in $\mathcal{ALC}$. During the construction process, a refutation graph is used as the guide to generate explanations.

Introduction

Description logics (DLs) have long been considered by researchers in Knowledge Representation and Reasoning. They have a wide range of applications such as domain modelling, software engineering, configuration, and the semantic web (Baader & Nutt 2003). However, existing DL reasoners, such as Racer (Haarslev & Möller 2001), do not provide users explanation services; they merely answer "Yes" or "No" to a satisfiability and/or consistency query with no further details about the sources. For applications with complex knowledge to represent and reason with, errors due to inconsistencies become quite common. For example, the DICE (Diagnoses for Intensive Care Evaluation) terminology (Schlobach & Cornet 2003) contains more than 2400 concepts, out of which about 750 concepts were unsatisfiable due to migration from other terminological systems. Although Racer can detect such inconsistencies and generate a list of unsatisfiable concepts, simply providing such a list is of little help for knowledge engineers and ontology developers to identify the sources of inconsistencies. Therefore, it is crucial to develop explanation services as an essential feature and useful tool for DL reasoners.

There are two options, in general, to build an explanation system. One is to construct a system specially designed to provide explanations, as suggested in (Wick & Thompson 1992). Several approaches on explanations in theorem proving systems (Huang 1994; Meier 2000) follow this idea. The other way is to add an explanation module to the reasoner so that reasoning procedures can be traced to generate explanations. Most research on explanations in logic programming and deductive databases (Byrd 1980; Shmueli & Tsur 1990; Arora *et al.* 1993) follow this approach.

In this paper, we propose to use resolution proofs to construct explanations for DL reasoners, which mainly follows the idea of the first approach mentioned above, while the necessary interaction with the DL reasoner is also considered. The reason for our preference of the first option is that most implemented DL reasoners use tableau algorithms as the underlying reasoning calculus. Tableau rules are designed to render the results faster but not necessarily easier for the users to understand. For instances, for a subsumption query, a tableau based reasoner first negates the subsumer, conjuncts it with the subsumee, and then tries to construct a model. The subsumption relationship holds if such a model cannot be found. However, it would be inappropriate to argue "$A \sqsubseteq B$ since $A \sqcap \neg B$ is not satisfiable" in an explanation. Besides, one possible unsatisfying explanation of unsatisfiability, especially in the context of disjunctions, is that every source is tried out but failed. Even if this is extended to include a trace of what was tried and why it failed, these traces may include many deduction paths and quickly become unwieldy. Furthermore, some DL optimization techniques, such as absorption[1], are adopted to make reasoning more efficient, but they are difficult to understand if presented as explanations to general users. Hence in DL a faithful trace of the reasoning is not a good explanation.

It is acknowledged by both DL and first-order logic (FOL) communities that standard methods of automated theorem proving in FOL are not suitable for DL tasks, especially if a significant number of negative tests is involved (Hustadt & Schmidt 1999). However, FOL proof procedure such as resolution can shed light on how explanations of DL reasoning can be formulated. Compared to natural deduction proofs or tableau proofs, the resolution technique is more focused, as all the literals involved in a proof contribute directly to the solution. Moreover, in the framework of resolution proofs, reasoning with global axioms and ABoxes is not more complex than reasoning about concept expressions alone, in contrast to the complexity of reasoning for most description logics. Below is a natural deduction proof style

*This work was supported in part by Natural Sciences and Engineering Research Council (NSERC) of Canada and by ENCS, Concordia University.

[1]The basic idea of absorption is to transform a general axiom, e.g., $C \sqsubseteq D$, to the form of a primitive definition $A \sqsubseteq D'$, where A is an atomic concept name and C may be a non-atomic concept.

explanation of a variant of the example in (Huang 1996). Suppose A and $A \sqsubseteq B$ are the premises, and B is the conclusion. The first column represents the line number. The second represents the logical hypotheses on which a line depends on. The third column is the conclusion formulas. The last column represents the inference rule that justifies a line, followed by the premise lines.

No	Hypothesis	Formula	Reason
1.	1	$\vdash A$	(Hypothesis)
2.	2	$\vdash A \sqsubseteq B$	(Hypothesis)
3.	3	$\vdash \neg B$	(Hypothesis)
4.	2	$\vdash \neg A \vee B$	(Tautology 2)
5.	5	$\vdash \neg A$	(Hypothesis)
6.	1, 5	$\vdash \bot$	($\neg E$ 1 5)
7.	7	$\vdash B$	(Hypothesis)
8.	3,7	$\vdash \bot$	($\neg E$ 3 7)
9.	1, 2, 3	$\vdash \bot$	(Case analysis 4 6 8)
10.	1, 2	$\vdash B$	(Indirect 9)

Considering that this problem can be proved with two steps of resolution as shown below, with the other columns representing the same thing as above and the third column representing the clauses of the conclusions, the natural deduction proof is indeed long and somewhat tedious.

No	Hypothesis	Clause	Reason
1.	1	$\{A\}$	(Hypothesis)
2.	2	$\{\neg A, B\}$	(Hypothesis)
3.	3	$\{\neg B\}$	(Hypothesis)
4.	1, 2	$\{B\}$	(Resolution 2 3)
5.	3, 4	$\bot$	(Resolution 3 4)

Most existing explanation facilities in resolution based automated theorem proving (ATP) transform the proofs into natural language style explanations (Lingenfelder 1996; Huang 1994; Meier 2000). They are specifically designed to solve problems in theorem proving, particularly in mathematics. Since ATPs focus on proving conclusions using theorems, lemmas and premises, their approach is not suitable for indirect proofs. Our approach uses resolution proofs for explanations of DL reasoning, especially unsatisfiability and inconsistency reasoning, with a goal different than others in ATP.

The rest of this paper is organized as follows. First we provide a brief introduction to Description Logics as a background. Then we presents the main components of the framework, and provides the algorithms for generating explanations. It also includes examples to illustrate the procedure to construct explanations. At last we conclude with some open issues and future work.

Background

Description logics are a family of concept-based knowledge representation formalisms. It represents the knowledge of a domain by first defining the relevant concepts of the domain. These concepts are used to specify properties of the objects and individuals in the domain. Typically a DL language has two parts: *terminology* (TBox) and *assertion* (ABox). The TBox includes intensional knowledge in the form of axioms whereas the ABox contains the extensional knowledge that is specific to elements in the domain, called individuals.

Among DL frameworks, $\mathcal{ALC}$ ($\mathcal{AL}$ stands for *Attribute language* and $\mathcal{C}$ stands for *Complement*) has been considered as a basic DL language of interests in numerous studies in DL. In $\mathcal{ALC}$ and other DL languages as well, basic descriptions are *atomic concepts*, designated by unary predicate symbols, and *atomic roles*, designated by binary symbols and used to express relationships between concepts. Arbitrary concept descriptions such as C and D are built from atomic concepts and roles recursively according to the following rules:

$$
\begin{array}{rl}
C, D \rightarrow A| & \text{(atomic concept)} \\
\neg C| & \text{(arbitrary concept negation)} \\
C \sqcap D| & \text{(intersection)} \\
C \sqcup D| & \text{(union)} \\
\forall R.C| & \text{(value restriction)} \\
\exists R.C & \text{(existential quantification)}
\end{array}
$$

where A denotes an atomic concept and R denotes an atomic role. The *intersection* (or *union*) of concepts, which is denoted $C \sqcap D$ (or $C \sqcup D$), is used to restrict the individuals to those that belong to both C and D (or either C or D). The *value restriction*, denoted $\forall R.C$, requires that all the individuals that are in the relationship R with the concept being described belong to the concept C. The *existential quantification*, written $\exists R.C$, shows that there must exist an individual that is in the relationship R with the concept being described and belongs to the concept C. The *universal concept* $\top$ is a synonym of $A \sqcup \neg A$. The *bottom concept* $\bot$ is a synonym of $A \sqcap \neg A$.

An interpretation $\mathcal{I}$ defines a formal semantics of concepts and individuals in $\mathcal{ALC}$. It consists of a non-empty set $\Delta^{\mathcal{I}}$, called the domain of the interpretation, and an interpretation function, which maps every atomic concept A to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and maps every atomic role R to a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. In addition, $\mathcal{I}$ maps each individual name a to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The interpretation $\mathcal{I}$ is extended to concept descriptions, as shown in Table 1.

Constructors	Semantics
A	$A^{\mathcal{I}}$
$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\forall R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$
$\exists R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$
$a : A$	$a^{\mathcal{I}} \in A^{\mathcal{I}}$
$(a,b) : R$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

Table 1: Interpretations of DL constructors in $\mathcal{ALC}$.

Axioms express how concepts and roles are related to each other. Generally, an axiom is a statement of the form $C \sqsubseteq D$ or $C \equiv D$, where C and D are concept descriptions.

An interpretation $\mathcal{I}$ satisfies $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. It satisfies $C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$.

The basic inference services in TBoxes include satisfiability, subsumption, equivalence, and disjointness. A concept in a TBox $\mathcal{T}$ is said to be *satisfiable* w.r.t $\mathcal{T}$ if there exists an interpretation $\mathcal{I}$ that is a model of $\mathcal{T}$. A model for $\mathcal{T}$ is an interpretation that satisfies it. The other three inference services can all be reduced to (un)satisfiability. Another important reasoning service in TBoxes is to check whether a TBox $\mathcal{T}$ is *consistent*, i.e., whether there exists a model for $\mathcal{T}$. The basic reasoning tasks in ABoxes are instance check, realization, and retrieval. The instance check verifies if a given individual is an instance of a specified concept. The realization finds the most specific concept that an individual is an instance of. The retrieval finds the individuals in the knowledge base that are instances of a given concept. An ABox $\mathcal{A}$ is *consistent* w.r.t a TBox $\mathcal{T}$, if there is an interpretation that is a model of both $\mathcal{A}$ and $\mathcal{T}$. Similar to the inference services in TBoxes, the other three inference services in ABoxes can also be reduced to the consistency problem of ABoxes. Further details of description logics can be found in (Baader & Nutt 2003).

Resolution Based Framework

Our explanation procedure consists of three components, described as follows. The architecture of the system is shown in Figure 1.

1. Preprocessing: The explanation system communicates with a DL reasoner, e.g., Racer, and provides the answer to a query, normally in the form of "Yes" or "No" for a concept satisfiability, or an ABox consistency query, or a list of unsatisfiability concepts for a TBox consistency checking. If the answer is "No", i.e., a concept is unsatisfiable or a TBox/ABox is inconsistent, together with the original TBox/ABox, they will be translated into FOL formulas or clauses by the translation component.
2. Rendering resolution proofs: A resolution based automated theorem prover is used to generate resolution proofs, taking the translated formulas or clauses as inputs. It is important to note that since a DL reasoner is used first to get the result of the query, hence the unsatisfiability of the concept or the inconsistency of the TBox/ABox is already known. The complexity of the resolution based decision procedure of the guarded fragment of FOL is double exponential in the worst case as shown in (Ganzinger & de Nivelle 1999). Since $\mathcal{ALC}$ can be embedded into the guarded fragment and decided by (Ganzinger & de Nivelle 1999), theoretically the termination is guaranteed.
3. Explaining: The resolution proof is sent back to the explanation kernel to be reconstructed for better human understanding. Our approach transforms the proof into its corresponding refutation graph (Eisinger 1991), a more abstract representation, for reconstruction of the explanation. A refutation graph is a graph whose nodes are literals (grouped together in clauses) and its edges connect complementary literals, corresponding to each resolution step in the proof. As shown in (Eisinger 1991), for each resolution proof, there is a minimal refutation graph, but one refutation graph can represent multiple different sequences of resolution steps. Since one of these sequences may have better explanation quality over others, we can dynamically choose among them to find the one with the best outcome. At last the clauses involved in each resolution step are traced back to the contributing DL axioms/assertions or FOL formulas and later transformed into natural language explanations if necessary.

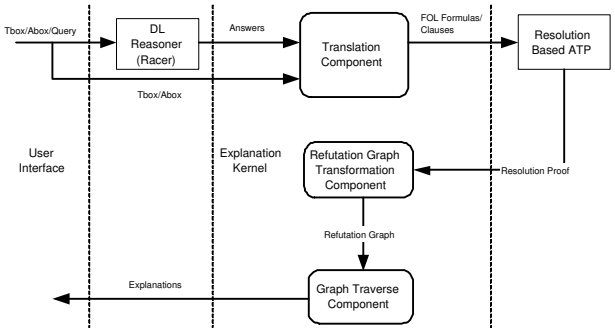

Figure 1: The system architecture.

Let us now consider two examples illustrating the applicability of our framework. The first example is a TBox as follows. We can find out that $A1$ is unsatisfiable, because if $A1$ is satisfiable, then $A2 \sqcap \neg A$ is satisfiable, consequently $A \sqcap \neg A$ is satisfiable, which obviously is a contradiction.

1. $A1 \sqsubseteq A2 \sqcap \neg A \sqcap A3$
2. $A2 \sqsubseteq A \sqcap A4$

The second example is an inconsistent ABox w.r.t an empty TBox. Intuitively, we can see the contradiction is due to i being an instance of $\forall$ S.$\neg$D as a known fact, but at the same time i belonging to $\exists$ S.D because k: $\forall$ R.$\forall$ R.$\exists$ S.D, (k, j): R and (j, i): R.

1. i: $\forall$ S. $\neg$D	2. (i, j): R
3. (j, i): R	4. (j, k): R
5. (k, j): R	6. k: $\forall$ R.$\forall$ R.$\exists$ S.D

In the following sections, we will show how such explanations are generated mechanically in our framework.

Translation between DL and FOL

The translation between DL and FOL is straightforward based on the semantics of DL. For $\mathcal{ALC}$, concepts can be translated into the first order predicate logic over unary and binary predicates with 2 variables, say x, y, which is denoted as $\mathcal{L}^2$. Table 2 shows such a translation from $\mathcal{ALC}$ into $\mathcal{L}^2$. An atomic concept A is translated into a predicate logic formula $\phi_A(x)$ with one free variable x such that for every interpretation $\mathcal{I}$, the set of elements of $\Delta^{\mathcal{I}}$ satisfying $\phi_A(x)$ is exactly $A^{\mathcal{I}}$. Similarly, a role name R can be translated into binary predicate $\phi_R(x, y)$. An individual name a is translated into a constant a. More details of the translation can be found in (Baader & Nutt 2003).

DL Constructor	FOL Formula
A	$\phi_A(x)$
$\neg C$	$\neg\phi_C(x)$
$C \sqcap D$	$\phi_C(x) \wedge \phi_D(x)$
$C \sqcup D$	$\phi_C(x) \vee \phi_D(x)$
$C \sqsubseteq D$	$\forall x(\phi_C(x) \rightarrow \phi_D(x))$
$\exists R.C$	$\exists y(\phi_R(x,y) \wedge \phi_C(y))$
$\forall R.C$	$\forall y(\phi_R(x,y) \rightarrow \phi_C(y))$
$a : A$	$\phi_A(a)$
$(a,b) : R$	$\phi_R(a,b)$

Table 2: Translation from $\mathcal{ALC}$ into $\mathcal{L}^2$.

In the circumstances of TBoxes inconsistency queries, there are two possible scenarios: a set of unsatisfiable concepts or an inconsistent TBox, i.e., the whole TBox is inconsistent thus causing all the concepts in the TBox to be unsatisfiable. In the context of ABoxes inconsistency queries, there are two possible situations: an individual is asserted as an instance of an unsatisfiable concept or it is asserted to belong to an inconsistent concept description. Consequently, these cases are distinguished in the translation.

Definition 1 Let $\mathcal{T}$ be a TBox, and C be an unsatisfiable concept in $\mathcal{T}$. An *unsatisfiability translation function* Γ maps $\mathcal{T} \cup \{C\}$ into the corresponding set of FOL formulas.

For example, suppose $\mathcal{T} = \{C \sqsubseteq B \sqcap \neg B\}$. C is unsatisfiable, so C and the axiom in $\mathcal{T}$ form the input set of Γ.

Definition 2 Let $\mathcal{T}$ be a TBox and $\mathcal{A}$ be an ABox (either $\mathcal{T}$ or $\mathcal{A}$ can be empty). An *inconsistency translation function* Λ maps $\mathcal{T} \cup \mathcal{A}$ into the corresponding set of FOL formulas.

For example, suppose TBox $\mathcal{T}$ consists of $\{C \sqsubseteq B \sqcap \neg B, A \sqcup \neg A \sqsubseteq C\}$. These two axioms form the input set of Λ.

The first example can be translated into FOL formulas and clauses as illustrated below, where $\mathbf{R_i}$ is the line number of the FOL formulas and $\mathbf{C_i}$ is the line number of the clauses, with $i = 1, \ldots, n$ representing the line number of the corresponding DL axiom.

$\mathbf{R_0}$. $\exists x A1(x)$.
$\mathbf{R_1}$. $\forall x A1(x) \rightarrow A2(x) \wedge \neg A(x) \wedge A3(x)$.
$\mathbf{R_2}$. $\forall x A2(x) \rightarrow A(x) \wedge A4(x)$.

$\mathbf{C_0}$ =$\{A1(c)\}$
$\mathbf{C_{11}}$ = $\{\neg A1(x), A2(x)\}$
$\mathbf{C_{12}}$ = $\{\neg A1(x), \neg A(x)\}$
$\mathbf{C_{13}}$ = $\{\neg A1(x), A3(x)\}$
$\mathbf{C_{21}}$ = $\{\neg A2(x), A(x)\}$
$\mathbf{C_{22}}$ = $\{\neg A2(x), A4(x)\}$

Since $A1$ is unsatisfiable, all the FOL formulas $\mathbf{R_i}$ or the clauses $\mathbf{C_i}$, $i = 1, \ldots, n$, and the unsatisfiable concept $A1$ form the input set of the unsatisfiability translation function Γ.

Similarly, since the second example is an ABox inconsistency query, all the FOL formulas $\mathbf{R_i}$ or the clauses $\mathbf{C_i}$, $i = 1, \ldots, n$, form the input set of the inconsistency translation function Λ. The corresponding FOL formulas and clauses after the translation are as following, with $f(y,z)$ symbol being result of Skolemization.

$\mathbf{R_1}$. $\forall y\ (S(i,y) \rightarrow \neg D(y))$.
$\mathbf{R_2}$. $R(i,j)$.
$\mathbf{R_3}$. $R(j,i)$.
$\mathbf{R_4}$. $R(j,k)$.
$\mathbf{R_5}$. $R(k,j)$.
$\mathbf{R_6}$. $\forall y \forall z \exists x\ (R(k,y) \rightarrow (R(y,z) \rightarrow S(z,x) \wedge D(x)))$.

$\mathbf{C_1}$ =$\{\neg S(i,y), \neg D(y)\}$
$\mathbf{C_2}$ =$\{R(i,j)\}$
$\mathbf{C_3}$ = $\{R(j,i)\}$
$\mathbf{C_4}$ =$\{R(j,k)\}$
$\mathbf{C_5}$ = $\{R(k,j)\}$
$\mathbf{C_{61}}$ = $\{\neg R(k,y), \neg R(y,z), D(f(y,z)\}$
$\mathbf{C_{62}}$ = $\{\neg R(k,y), \neg R(y,z), S(z,f(y,z))\}$

Refutation graph

In ordinary resolution proofs, the order in which a sequence of resolutions is done can have a significant impact on the result. For example, we can obtain two different proof trees in the first example by rearranging the orders of the resolving clauses, as shown in Figure 2. For the sake of simplicity, variables are omitted in the figure. The difference between these two proofs is due to the choice of resolving $\{\neg A2, A\}$ with $\{\neg A\}$ or $\{A2\}$ first. Obviously, Proof 1 is better for human understanding, because later when original DL axioms of each resolution step are traced back to render explanations, the axiom $A2 \sqsubseteq A$ is more appropriate to be used as an inference rule to explain A is the consequence of $A2$ than $\neg A2$ is the consequence of $\neg A$. If resolution proofs are transformed into refutation graphs, it does not mater in which order the resolutions are done, as the resulting refutation graph can be made the same. The final refutation graph can be built using the resolving ordering of Proof 1 in Figure 2, but the explanations can be constructed by the resolving order of Proof 2.

We will use the following definitions of refutation graphs in our presentation. See (Eisinger 1991) for more details.

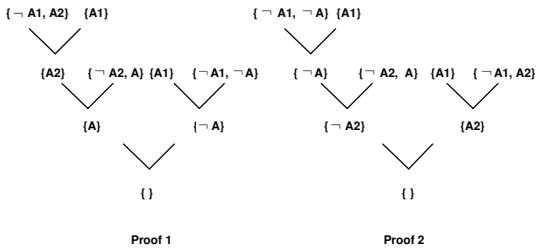

Figure 2: Two different results from resolutions on Ex.1.

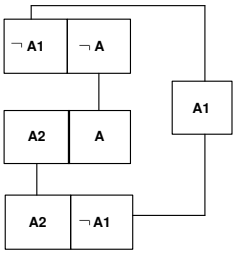

Figure 3: The refutation graph of Ex.1.

Definition 3 A *refutation graph* is a quadruple $\mathcal{G} = (\mathcal{L}, \mathcal{C}, \mathcal{M}_{\mathcal{L}}, \mathcal{K})$, where $\mathcal{L}$ is a finite set of literal nodes in $\mathcal{G}$, $\mathcal{C}$ is a partition of the set of literal nodes and its members are clause nodes in $\mathcal{G}$. $\mathcal{M}_{\mathcal{L}}$ is a mapping from $\mathcal{L}$ to a set of literals, which labels the literal nodes with literals. The set of links $\mathcal{K}$ is a partition of a subset of $\mathcal{L}$. All the literal nodes in one link are labeled with literals which are unifiable. There are no pure literal nodes in a refutation graph, i.e., every literal node belongs to some link in $\mathcal{K}$.

The refutation graph of the first example is shown in Figure 3. We extend the above notions to explain DL reasoning.

Definition 4 A *labeled refutation graph* is a quintuple $\mathcal{G}' = (\mathcal{L}, \mathcal{C}, \mathcal{M}_{\mathcal{L}}, \mathcal{K}, \mathcal{M}_{\mathcal{D}})$. $\mathcal{M}_{\mathcal{D}}$ is a mapping from $\mathcal{L}$ to a FOL formula, which labels the literal nodes with their originating FOL formulas. Other symbols have the same meaning as in the definition of a refutation graph.

In the first example, one of the mappings in $\mathcal{M}_{\mathcal{D}}$ is $\{\neg A1(x)\} \mapsto \forall\, x\; A1(x) \rightarrow A2(x) \wedge \neg A(x) \wedge A3(x)$.

Definition 5 A *traversing path* of the labeled refutation graph involving clause nodes $C_1, \ldots, C_n$ is a sequence (M_1, C_1), (L_2, C_2), (M_2, C_2), (L_3, C_3), (M_3, C_3), $\ldots$, (L_n, C_n), where L_i and M_i, $i = 1, \ldots, n$, are literal nodes in C_i and $L_i \neq M_i$. Also for all $i < n$, M_i and L_{i+1} unify, i.e., there is a link between them.

Definition 6 A *preferred literal node* is a literal node that is originated from a unit clause or the left hand side of an implication in FOL formulas translated from original DL axioms. A literal node associated with a preferred literal node is the corresponding right hand side of the implication. For example, in the FOL formula $\forall x(A(x) \rightarrow (B(x) \rightarrow C(x)))$, literal nodes $\neg B(x)$ and $C(x)$ are associated with $\neg A(x)$ and node $C(x)$ is associated with $\neg B(x)$.

In the first example, the preferred literal nodes are $\neg A1(x)$ and $\neg A2(x)$.

Explanation based on labeled refutation graphs

The main idea of explanations based on the labeled refutation graph is to start from a literal node (or nodes) and traverse the graph. The traversal order is decided by a preference assigned to each node. For example, the axiom $A \sqsubseteq B$ is more appropriate to be used as an inference rule to explain B is the consequence of A than $\neg A$ is the consequence of $\neg B$. Therefore, accordingly there is a preference on A instead of B, i.e., the literal node $\neg A$ should be traversed before B is traversed. When a literal node is being traversed, its label is added to the explanation list. The procedure terminates if the graph traversal is complete. After the traversal is completed, each label is translated into an entry in an explanation list consisting of its source axioms in DL or formulas in FOL. After some clean-up, e.g., deleting duplicates, this explanation list can be further transformed into natural language style explanations.

Explaining unsatisfiable concepts The algorithm in Figure 4 provides explanations for unsatisfiability queries. It uses a stack, called SOT, which includes the labeled literal nodes which are yet to be traversed. It also uses a queue data structure, called QOT, which includes the labels of already traversed nodes.

UnsatisfiableTraverse
(labeled refutation graph $\mathcal{G}'$, the unsatisfiable concept $\mathcal{C}$)

push the associated literal node of $\mathcal{C}$ into SOT
for all the literal nodes L_i in SOT
  put the labels of L_i into QOT
  for all L_j such that there is a link between L_i and L_j,
    L_j is a preferred literal node
    for all the unpreferred literal nodes L_k that are
      associated with L_j
      push L_k into SOT
    put the labels of L_j into QOT
  pop L_i from SOT
return QOT

Figure 4: Unsatisfiable concept traverse algorithm.

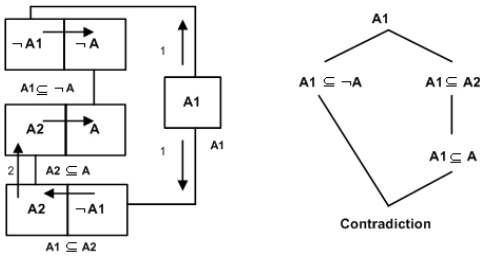

Figure 5: The refutation graph and its explanation of Ex.1.

The refutation graph and the traversal order of the first example are shown on the left in Figure 5, with the number on the link denoting the traversal order, the arrow indicating the traversal direction, and the axiom below each clause node indicating its corresponding label. For the sake of simplicity, variables are omitted. The graph on the right of this figure shows the labels organized as explanations after the traversal and its explanation list in FOL formulas form is shown as bellow:

$\mathbf{L_1}$: $A1(c)$, $A1(x) \rightarrow A2(x)$ $\quad \Rightarrow A2(c)$
$\mathbf{L_2}$: $A1(c)$, $A1(x) \rightarrow \neg A(x)$ $\quad \Rightarrow \neg A(c)$
$\mathbf{L_3}$: $A2(x) \rightarrow A(x)$, $A2(c)$ $\quad \Rightarrow A(c)$

where $\mathbf{L_i}$, $i = 1, \ldots, n$, is the line number of the list and the comma on the left hand side denotes conjunction.

Explaining inconsistent TBoxes/ABoxes For inconsistency queries, it is difficult to know where to begin the traversal since the whole TBox/ABox is inconsistent. This is where resolution proofs come into the picture. All the literal nodes involved in the first step of the resolution proof will be put into the starting set. The algorithm is shown in Figure 6, with SOT and QOT designating the yet-to-be traversed literal nodes which are yet-to-be traversed nodes and already traversed nodes.

InconsistencyTraverse
(labeled refutation graph $\mathcal{G}'$)

push the preferred literal nodes involved in the first step of the resolution proof into SOT
push the grounded unit clause nodes into SOT
for all the literal nodes L_i in SOT
    put the label of L_i into QOT
    for all the literal nodes L_j that has a link with L_i,
        L_j is a preferred node
        for all the unpreferred literal nodes L_k that
            are associated with L_j
            push L_k into SOT
        put the labels of L_j into QOT
    pop L_i from SOT
return QOT

Figure 6: Inconsistent TBoxes/ABoxes traverse algorithm.

In our second example, the first step of the resolution proof involves $\{\neg R(k,y),\ \neg R(y,z),\ D(f(y,z)\}$,$\{R(j,i)\}$ and $\{R(k,j)\}$. The literal nodes of these clauses form the traversal starting set. The traversal procedure is shown in Figure 7.

The following is the explanation list after the traversal of our example terminates. It can be translated into a human understandable explanation: Since $R(k,j)$, $R(j,i)$ and for all y and z such that $R(k,y)$ and $R(y,z)$ there exists a x that $R(z,x)$ and $D(x)$ hold, so $D(x)$ must be true. Similarly, we can conclude $S(i,x)$. But for all i, the right hand side of S must be not D. A contradiction arises, hence the ABox is inconsistent.

$\mathbf{L_1}$: $\forall y \forall z \exists x\ (R(k,y) \rightarrow (R(y,z) \rightarrow S(z,x) \wedge D(x))),$
$R(k,j), R(j,i) \Rightarrow S(i,f(j,i))$
$\mathbf{L_2}$: $\forall y \forall z \exists x\ (R(k,y) \rightarrow (R(y,z) \rightarrow S(z,x) \wedge D(x))),$
$R(k,j), R(j,i) \Rightarrow D(f(j,i))$
$\mathbf{L_3}$: $S(i,f(j,i)), \forall y\ S(i,y) \rightarrow \neg\ D(y) \Rightarrow \neg D(f(j,i))$

Related Work

There have been several proposals to provide explanations for DL reasoning. The earliest work is (McGuinness & Borgida 1995) which provides an explanation facility for subsumption and non-subsumption reasoning in CLASSIC (Brachman *et al.* 1990). CLASSIC is a family of knowledge representation systems based on description logics and it allows universal quantification, conjunction and restricted number restrictions. In (McGuinness & Borgida 1995),

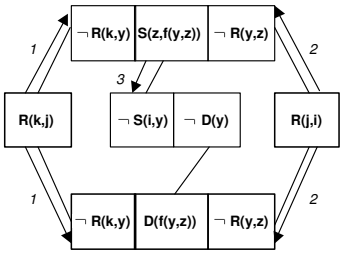

Figure 7: The traversal procedure of Ex.2.

they propose to explain in a proof-theoretic manner based on structural subsumption, using inference rules to perform structural subsumption comparisons. Lengthy explanations are decomposed into smaller steps and a single step explanation is followed by more detailed explanations. This work was extended in (Borgida *et al.* 1999) by using sequent rules to explain subsumption. The sequent rules are modified to imitate the behavior of tableau calculus as well as the behavior of human reasoning. In contrast to these works, (Schlobach & Cornet 2003) provides algorithms to pinpoint the unsatisfiable concepts and related axioms.

Conclusion and Future work

In this paper, we proposed a resolution proof based framework for explaining DL reasoning. We have developed the basic algorithms to generate explanations and demonstrated a first step towards the applicability of this framework to explain unsatisfiability and inconsistency queries w.r.t TBoxes/ABoxes in $\mathcal{ALC}$. An implementation of the proposed framework is underway.

Currently, this approach is still in a premature status and we are working on the following limitations:

1. The algorithms for traversing the labeled refutation graph need to be refined and more closely studied, especially the algorithm to explain inconsistent TBoxes and ABoxes. Proofs of their soundness and completeness should also be presented.
2. To ensure that the first-order logic prover terminates in reasonable time, more sophisticated translations from DL axioms/assertions into FOL formulas should be used, but then there will be a gap between the clauses used during the resolution and the axioms in the DL knowledge base. Besides, skolemization also blurs the relation between the resulting clauses and its original DL axioms, which may introduce complications for explanations. One possible solution is to eliminate skolem functions from resolution proofs as suggested in (de Nivelle 2003).
3. Currently the explanation is restricted to unsatisfiability and inconsistency queries. Although we believe explanations for this kind of queries are more useful for general users to debug their terminologies, providing explanations for satisfiability and non-subsumption queries is also necessary. As pointed out in (McGuinness & Borgida 1995), an explanation for satisfiability and non-subsumption queries can be generated by returning a

model or a counter-example but more work needs to be done.

References

Arora, T.; Ramakrishnan, R.; Roth, W. G.; Seshadri, P.; and Srivastava, D. 1993. Explaining program execution in deductive systems. In *Deductive and Object-Oriented Databases*, 101–119.

Baader, F., and Nutt, W. 2003. Basic description logic. In Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. F., eds., *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press. 5–44.

Borgida, A.; Franconi, E.; Horrocks, I.; McGuinness, D. L.; and Patel-Schneider, P. F. 1999. Explaining $\mathcal{ALC}$ subsumption. In *1999 International Workshop on Description Logics (DL1999)*.

Brachman, R. J.; McGuiness, D. L.; Patel-Schneider, P. F.; and Resnick, L. A. 1990. Living with CLASSIC: when and how to use a KL-ONE-like language. In Sowa, J., ed., *Principles of semantic networks*. San Mateo, US: Morgan Kaufmann.

Byrd, L. 1980. Understanding the control flow of prolog programs. *Proceedings of the 1980 Logic Programming Workshop* 127–138.

de Nivelle, H. 2003. Translation of resolution proofs into short first-order proofs without choice axioms. In Baader, F., ed., *Proceedings of the 19th International Conference on Computer Aided Deduction (CADE 19)*, volume 2741 of *Lecture Notes in Artificial Intelligence*, 365–379. Miami, USA: Springer Verlag. An extended version will appear in a special issue of selected CADE papers.

Eisinger, N. 1991. *Completeness, Confluence, and Related Properties of Clause Graph Resolution*. Morgan Kaufmann Publishers Inc.

Ganzinger, H., and de Nivelle, H. 1999. A superposition decision procedure for the guarded fragment with equality. In *LICS '99: Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science*, 295. Washington, DC, USA: IEEE Computer Society.

Haarslev, V., and Möller, R. 2001. Racer system description. In R. Gori, A. Leitsch, T. N., ed., *Proceedings of International Joint Conference on Automated Reasoning, IJCAR'2001*, 701–705. Siena, Italy: Springer-Verlag.

Huang, X. 1994. Reconstructing proofs at the assertion level. In Bundy, A., ed., *Proc. 12th Conference on Automated Deduction*, 738–752. Springer-Verlag.

Huang, X. 1996. Translating machine-generated resolution proofs into nd-proofs at the assertion level. In Foo, N. Y., and Goebel, R., eds., *PRICAI*, volume 1114 of *Lecture Notes in Computer Science*, 399–410. Springer.

Hustadt, U., and Schmidt, R. A. 1999. Issues of decidability for description logics in the framework of resolution. In Caferra, R., and Salzer, G., eds., *Automated Deduction in Classical and Non-Classical Logics*, volume 1761 of *LNAI*, 192–206. Springer.

Lingenfelder, C. 1996. *Transformation and structuring of computer generated proofs,*. Ph.D. Dissertation, University of Kaiserslautern.

McGuinness, D. L., and Borgida, A. 1995. Explaining subsumption in description logics. In *Proceedings of the tenth International Joint Conference on Artificial Intelligence, IJCAI'95*.

Meier, A. 2000. TRAMP: Transformation of Machine-Found Proofs into Natural Deduction Proofs at the Assertion Level. In McAllester, D., ed., *Proceedings of the 17th Conference on Automated Deduction (CADE–17)*, volume 1831 of *LNAI*, 460–464. Pittsburgh, USA: Springer Verlag, Berlin, Germany.

Schlobach, S., and Cornet, R. 2003. Non-standard reasoning services for the debugging of description logic terminologies. In *Proceedings of the eighteenth International Joint Conference on Artificial Intelligence, IJCAI'03*. Morgan Kaufmann.

Shmueli, O., and Tsur, S. 1990. Logical diagnosis of ldl programs. In Warren, D. H. D., and Szeredi, P., eds., *Logic Programming: Proc. of the Seventh International Conference*. Cambridge, MA: MIT Press. 112–129.

Wick, M. R., and Thompson, W. B. 1992. Reconstructive expert system explanation. *Artif. Intell.* 54(1-2):33–70.