Rewriting Rules into $\mathcal{SROIQ}$ Axioms

Francis Gasse[1], Ulrike Sattler[2] and Volker Haarslev[1]

[1] Concordia University, Montreal, Quebec, Canada
{ f_gasse,haarslev } @cse.concordia.ca,
[2] University of Manchester, UK
sattler@cs.man.ac.uk

Abstract. Description Logics are a family of very expressive logics but some forms of knowledge are much more intuitive to formulate otherwise, say, as rules. Rules in DL can be dealt with two approaches: (i) use rules as they are knowing that it leads to undecidability. (ii) or make the rules DL-safe, which will restrict their semantic impact and, e.g., loose the nice "car owners are engine owners" inference. Here, we offer a third possibility: we rewrite the rule, if it satisfies certain restrictions, into a set of axioms which preserves the nice inferences. In this paper, we describe the rewriting technique and prove that it does really preserve the semantics of the rule. We have implemented the rewriting algorithm and have practical results.

1 Introduction

Recent advances in very expressive description logics [1] have made this family of logics capable of modeling some very complex knowledge. However, some forms of knowledge are easier to formulate as rules even if they are expressible as DL axioms. To deal with such rule encoded knowledge, one has two options: 1) either use it as is with, say, Hoolet [2], knowing that it leads to undecidability. 2) or "$\mathcal{O}$(x)-ify" it to make it DL-safe [3], but then we restrict its semantic impact and, e.g., loose the nice "car owners are engine owners" inference. In this paper we present a technique to rewrite a restricted form of tree-shaped rules into DL axioms using the features introduced in $\mathcal{SROIQ}$. The proposed technique has some very desirable characteristics, namely that (i) we can use standard DL reasoners, (ii) rules can be directly embedded into the knowledge base (KB) so we do not need a rule format standard, (iii) it does not require DL-safety to ensure termination and (iv) it can be used to convert existing SWRL [4] rules that satisfy the restrictions. However, since we only rewrite the rules, we do not add any expressivity to the logic but the rewriting uses $\mathcal{SROIQ}$'s constructs in a way that, to the best of our knowledge, has never been presented to encode rules in DL axioms.

Here are a few examples of the rules this rewriting can handle: (i)$R(x,y) \wedge R(x,z) \wedge C(y) \wedge D(z) \rightarrow C(x)$ (ii)$R(x,y) \wedge S(y,z) \rightarrow S(x,y)$ (iii)$\mathit{hasOffspring}(y,x) \wedge \mathit{hasParent}(x,y) \wedge \mathit{hasSibling}(y,z) \wedge \mathit{Man}(z) \rightarrow \mathit{hasUncle}(x,z)$. The rule (i) is trivial to express in an axiom and is thus obviously rewritable. The rules (ii) is also

trivially rewritable in $\mathcal{SROIQ}$. The rule (iii) is much more interesting because it is not obvious to rewrite, even for an experienced DL user. Hence the motivation of this paper.

2 Preliminaries

We will now introduce the basic concepts and notations used in this paper, in particular the DL $\mathcal{SROIQ}$, rules and graph conceptualization of rules.

2.1 The DL $\mathcal{SROIQ}$

$\mathcal{SROIQ}$ is the logical foundation of OWL 1.1 and some of the constructs it introduces are used extensively to rewrite the rules. We introduce only these features we use to rewrite the rules: the Self concept and the Role Inclusion Axiom (RIA). For more information on $\mathcal{SROIQ}$, the reader is referred to [5].

Complex RIA Complex role inclusion axioms are constructs of the form $R \circ S \sqsubseteq T$ where $\circ$ is a binary composition operator. For example[1], w.r.t. the axiom $owns \circ hasPart \sqsubseteq owns$, and the fact that each car contains an engine $Car \sqsubseteq \exists hasPart.Engine$, an owner of a car is also an owner of an engine, i.e., the following subsumption is implied: $\exists owns.Car \sqsubseteq \exists owns.Engine$. However, $\mathcal{SROIQ}$ requires the role hierarchy $\mathcal{R}_h$ to be regular. Regularity restricts the form of RIA allowed in $\mathcal{R}_h$ so we will introduce it. To define regularity, we first define a relation $\prec$ over the set of roles $\mathbf{R}\{R^- | R \in \mathbf{R}\}$ that is irreflexive, transitive and also satisfies $S \prec R \Longleftrightarrow S^- \prec R$ for all roles R and S. A RIA $l \dot{\sqsubseteq} R$ is $\prec$-regular if R is a role name and:

1. $l = RR$, or
2. $l = R^-$, or
3. $l = S_1, ..., S_n$ and $S_i \prec R$ for all $1 \leq i \leq n$, or
4. $l = RS_1, ..., S_n$ and $S_i \prec R$ for all $1 \leq i \leq n$, or
5. $l = S_1, ..., S_nR$ and $S_i \prec R$ for all $1 \leq i \leq n$.

Finally, a role hierarchy $\mathcal{R}_h$ is said to be regular if there exists a relation $\prec$ on roles such that each RIA in $\mathcal{R}_h$ is $\prec$-regular.

Self The $Self$ construct allows to express "local reflexivity", i.e., to model a role that has the same individual for antecedent and successor. For example[1], the concept "narcissist" can be defined as $\exists likes.Self$. The formal semantics of this construct are $(\exists R.Self)^{\mathcal{I}} = \{x | \langle x, x \rangle \in R^{\mathcal{I}}\}$.

[1] examples taken from [5]

2.2 Rules

We now formally define the rules and how they are interpreted w.r.t an interpretation.

Definition 1. *Let X be a set of variables disjoint from role or concept names. An* atom *is either a concept atom $C(x)$ or a role atom $R(x, y)$, for C a (possibly complex) $\mathcal{SROIQ}$ concept name, R a (possibly non-simple) role name and x, y variables from X. A* rule *is an expression of the form $A_1 \wedge A_2 \wedge ... \wedge A_n \rightarrow H$ where A_i and H are atoms of either form, and the variables in H occur in some of the A_i. Given a rule r, we use $head(r) = H$ for its head, $body(r) = \{A_1, \ldots, A_n\}$ for its body and $Var(r)$ for the variables it uses.*

In this definition, we have made three restrictions: (1) the variables in the head must occur in the body: this is the standard safety restrictions on rules (which is unrelated to DL-safety). (2) We only allow for role names in a rule, and not for inverse roles. This is without loss of generality since we can replace each $R^-(x, y)$ with $R(y, x)$ preserving its semantics.(3) We do not allow for individual names in a rule. Again, this is without loss of generality since we can, for example, replace each $C(a)$ with $C(x_a), \{a\}(x_a)$. We have made the latter two restrictions to make the following presentation easier—obviously, in our implementation, the user can write rules with inverse roles and individual names, and these replacements will be taken care of by our implementation.

Also, a rule has to be connected. A rule r is *connected* if there exists for all variables x,y in r a sequence $x_1....x_n$ such that $x_1 = x$ and $x_n = y$ and for all $1 \leq i < n$ there exists a role R such that $R(x_i, x_{i+1}) \in r$ or $Inv(R)(x_{i+1}, x_i) \in r$. From now on, when we use "rule", we implicitly mean "connected rule". We now introduce rule normalization, a technique defined as follows: for all roles R and S, such that $\mathcal{R} \models R \equiv S^-$, replace all atoms $S(x, y)$ in the rule with $R(y, x)$. We require the rules to be normalized to simplify the rewriting.

Definition 2. *Let $\mathcal{I}$ be an interpretation that is a model of a KB $\mathcal{K}$ and π be a* mapping *$Var(r) \rightarrow \Delta^{\mathcal{I}}$ where r is a rule. We write*

- $\mathcal{I} \models^{\pi} C(x)$ *if* $\pi(x) \in C^{\mathcal{I}}$
- $\mathcal{I} \models^{\pi} R(x, y)$ *if* $(\pi(x), \pi(y)) \in R^{\mathcal{I}}$

For a set of atoms Γ, we write $\mathcal{I} \models^{\pi} \Gamma$ if $\mathcal{I} \models^{\pi} at$ for all at in Γ. We say that $\mathcal{I}$ satisfies a rule r if, for all π, $\mathcal{I} \models^{\pi} body(r)$ implies that $\mathcal{I} \models^{\pi} head(r)$.

In this paper, we define the rewriting of a single rule into DL axioms w.r.t. a knowledge base. To extend it to a set of rules, we rewrite rules sequentially to avoid any inconsistent states. That follows from the non DL-safety of the rewritten rules, which implies that a rule can cause changes in either the concept or the role hierarchy. Also it is to be noted that for similar reasons, two rewritings of a given rule base might yield different results if the ordering of the rules changes.

2.3 Rules as Graphs

In this paper, we often refer to graph conceptualization of the rules. We do so because it allows us to state restrictions and algorithms in an elegant way. Furthermore, it is a well-known paradigm that is rather intuitive in this context. We will now introduce the relevant graph related notions. A *rule graph* is a directed labeled graph induced by the body of the rule. A labeled graph has the following structure $G = (V, E, \ell_V)$, with respect to two alphabets Σ_V and Σ_E (which in this context are respectively concept names and roles names), where:

- V is a finite set of nodes,
- $E \subseteq V \times \Sigma_E \times V$ is a ternary relation describing the edges (including the labeling of the edges) and
- $\ell_V : V \rightarrow \Sigma_V$ is a function describing the labeling of the nodes.

Given a rule r, its graph is defined as follows. For every concept atom of the form $C(x)$ in r, we have that $x \in V$ and $C \in \ell_x$. For every atom of the form $R(x, y)$ $(x \neq y)$, we have that $x, y \in V$ and $\langle x, R, y\rangle \in E$. For every atom of the form $R(x, x)$, we have that $x \in V$ and $\exists R.Self \in \ell_x$. The head of the rule is not included in its rule graph, but we keep track of the node(s) it references and keep it for later use. We now introduce the notion of a skeleton of a rule graph.

Definition 3. *A graph is a* skeleton *with respect to a role hierarchy $\mathcal{R}$ if there are no edges in E that are implied by other edges in E. We say that an edge $\langle x, R, y\rangle$ in a graph G is* implied*, if for all interpretation $\mathcal{I}$ for which we have $\mathcal{I} \not\models G$ we also have $\mathcal{I} \not\models G/\langle x, R, y\rangle$.*

Let us now explain the definition of an implied edge. If there exists an interpretation for which we don't have $\mathcal{I} \not\models G/\langle x, R, y\rangle$, it means that removing this edge undeconstrains the model, so it is not implied by the remaining edges. If no such interpretation exists, this edge is implied by the remaining ones. It is the only relevant case since any interpretation that models G trivially models any subgraph of G. The reason we use skeletons is that by removing implied edges from the rule graph we maximize the chance that it satisfies the restrictions we impose on the form of the rules (see next section). We now introduce a new relation, $\overline{\in}$, to simplify the writing of some expressions. The $\overline{\in}$ relation is an extension of the $\in$ relation in such a way that it covers the inverse elements. Let R be a role and x, y be variables. $\langle x, R, y\rangle \overline{\in} E$ if $\langle x, R, y\rangle \in E$ or $\langle y, Inv(R), x\rangle \in E$. Let G be a rule graph, x, y and z be elements of V_G and R and S be elements of Σ_{E_G}. A skeleton of G is obtained by applying the following rules to G until none is applicable anymore:

1. Let R be a transitive role w.r.t. $\mathcal{R}$. If $\langle x, R, y\rangle$, $\langle y, R, z\rangle$ and $\langle x, R, z\rangle \overline{\in} E$, then remove $\langle x, R, z\rangle$ from E.
2. Let R be a super-role of S w.r.t. $\mathcal{R}$. If $\langle x, R, y\rangle \overline{\in} E$, $\langle x, S, y\rangle \overline{\in} E$ and $R \neq S$ then remove $\langle x, R, y\rangle$ from E.
3. Let a RIA of the form $R_1 \circ R_2 \ldots R_n \sqsubseteq S$ be in $\mathcal{R}$, if $\langle x_1, R_1, x_2\rangle, \ldots, \langle x_n, R_n, x_{n+1}\rangle \overline{\in} E$ and $\langle x_1, S, x_{n+1}\rangle \overline{\in} E$, then remove $\langle x_1, S, x_{n+1}\rangle$.

All the skeletons of a graph are equivalent. This follows from the semantics of equivalent, transitive roles, role hierarchies and RIAs since we only remove the edges from the rule graph that are implied by the remaining ones according to their respective semantics. From now on, we will assume that all rule graphs are skeletons.

3 Admissible Rules

Before describing the rewriting process we must specify what type of rules are admissible to be rewritten. These restrictions follow from the encoding of the rules into $\mathcal{SROIQ}$. When dealing with multiple rules, they are added sequentially so that we ensure that they are admissible w.r.t. to each other (since a rule can modify the role hierarchy). Different conditions apply to the two types of rules so we will introduce them separately.

Definition 4. *A rule graph G is* concept-headed rule admissible *if, seen as an undirected graph, it is a tree, i.e., does not contain any cycle.*

Definition 5. *We introduce the* main chain *of a role-headed rule graph, it is the path between (inclusively) the two nodes referenced in the rule's head. A rule graph G, with role R as head, is* role-headed rule admissible *if:*

1. *It is a tree when seen as an undirected graph,*
2. *Let l be the list of roles of the main chain. The RIA $l \dot{\sqsubseteq} R$ must satisfy the restrictions mentioned in Section 2.1,*
3. *If R is the first(resp. last) role in the main chain, then the first(resp. last) node of the main chain does not contain any concept assertions and no other edges is connected to it.*

These definitions of admissible rules are complete, meaning that the rewriting of any rule not satisfying these restrictions would yield an invalid $\mathcal{SROIQ}$ KB and that all rules expressible in $\mathcal{SROIQ}$ satisfies these restrictions. The normalization and skeletonization steps ensure that varying the syntax of a rule does not alter the way it is treated.

4 Rewriting Process

We will now describe the rewriting technique and its different steps. The rewriting is composed of two steps : the graph folding, in which we convert nodes and edges into labels of other nodes so that only the nodes referenced in the head are left, and the rule insertion, in which we rewrite what is left of the rule graph into a DL axiom.

4.1 Graph folding

The rule graph has to be rolled-up before we insert the rule in the KB. The two types of rules have different forms of rolled-up graphs. The rolled-up graph of a concept-headed rule is a single node and the folded graph of a role-headed rule is the rule's main chain. First, we will describe the rolling-up technique and then how to roll-up each type of rule graph.

Roll-up a node This technique is widely used and known for DL, the reader may refer to [6] for more information, so we will only briefly present the technique to roll-up a single node into its ancestor, from which the general case of rolling up a tree is obvious. Given a rule tree-shaped graph G, to roll-up a leaf node, n_l, into its predecessor, n_p, is a very simple procedure. Let R be an element of Σ_E such that $\langle n_p, R, n_l \rangle \in E$. We just need to extend the label of n_p with the following $\sqcap \exists R.(\ell n_l)$ and remove n_l and the edge $\langle n_p, R, n_l \rangle$ from the graph.

Fold a concept-headed rule In this procedure, we use the notion of the root of a concept-headed rule's graph. The root is the node that is referenced in the head of the rule. The whole graph will be folded into that node. We do so simply by recursively rolling-up all the neighbors of the root into it.

Fold a role-headed rule The algorithm to fold a role-headed rule takes a rule graph G as input and its output is the graph folded into the main chain. We consider a graph S_G which is the subgraph of G obtained by removing the edges that are part of the main chain. By definition, each connected components of that graph will contain exactly one node that is part of the main chain. We just have to roll-up each connected components into the main chain's node it contains.

4.2 Rule insertion

The folded rule graph is now ready for insertion into the KB. Depending on the type of rule, we will insert either a GCI or a RIA. Let us now describe the procedure to construct these axioms.

Concept-headed rule The folded graph of a concept-headed rule is its root node. The label of that node is the rewriting of the rule's body, that is in fact the conjunction of the rolling-up of the root's neighbor(s). So the insertion of a rule is simply the addition of a GCI of the form $\ell_{root} \sqsubseteq C$, where ℓ_{root} is the label of the root node and C the concept of the head atom.

Role-headed rule The folded graph of a role-headed rule is its main chain. We create an RIA from it with the following procedure, where w is a role string that is initially empty. Steps 1.a and 1.b are referred to as "label removal".

1. For every node n(starting with the ancestor of the head atom)
   (a) Add the axiom $\ell_n \sqsubseteq \exists instC.Self$ to the KB, where $instC$ is a fresh role name and ℓ_n is the label of node n
   (b) Append the role $instC$ to the string w
   (c) Append the label of the edge leading to the successor of n to w
2. Add the RIA $w \sqsubseteq R$ to the KB, where R is the role in the head of the rule.

4.3 Example

In order to clearly illustrate how the rewriting works, we will now rewrite a rule step by step. The rewriting of concept-headed rules is quite intuitive so we will rewrite a role-headed rule. Since the "uncle" example was used so often to demonstrate the motivations to have rules integrated with DLs, we will use it for the example. Obviously, we can write a RIA to model this problem in $\mathcal{SROIQ}$, e.g., $hasParent \circ hasBrother \sqsubseteq hasUncle$, but we will assume that we do not have *hasBrother* to show the potential of the rewriting. Let us assume that the rule is formulated as follows: $hasOffspring(y, x) \wedge hasParent(x, y) \wedge hasSibling(y, z) \wedge Man(z) \rightarrow hasUncle(x, z)$. The rule graph of this rule is shown in Figure 1.

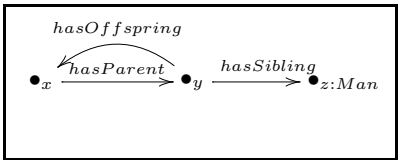

Fig. 1. The rule graph of the example

The first step is to make this rule graph a skeleton by removing implied edges. In our example KB, we have $hasParent \sqsubseteq hasOffspring^-$. It follows that the *hasOffspring* edge in the rule graph is implied by the *hasParent* edge, so the *hasOffspring* edge has to be removed from the graph. Now that the rule graph is a skeleton, we can proceed with the folding of the rule graph. However, since the folding of a role-headed rule graph consists of folding the subgraphs spawning from nodes of the main chain, this rule graph is already folded. We can now build the rule's axiom and insert it into the KB. For that we have to traverse the graph and build the role chain that we will add as a subrole of the head. Here is how to build a list of roles, w, for each nodes and edges:

- Node x:
  - No label in node.
  - Add *hasParent* to w.
- Node y:
  - No label in node.

  - Add *hasSibling* to w.
- Node z:
  - Add $Man \sqsubseteq \exists instMan.Self$ to the KB and *instMan* to w.
  - No successor.

It is to be noted that any label that would result from a rolling-up would be treated exactly the same way Man was. The content of w would then be $hasParent, hasSibling, instMan$. The last step is to insert the RIA modeling the rule in the KB, which is of the form $w \sqsubseteq R$ where R is the role in the head. The axiom for this example is $hasParent \circ hasSibling \circ instMan \sqsubseteq hasUncle$.

5 Correctness of the rewriting

The rewriting of a rule obviously requires the addition of axioms to the KB. In order to complete our proofs, we have to introduce the notion of an extension of an interpretation.

Definition 6. *$\mathcal{I}'$ is an* extension *of $\mathcal{I}$ if*

1. $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}'}$
2. *if* $\mathcal{I} \models \mathcal{K}$ *then* $\mathcal{I}' \models \mathcal{K}'$, *where* $\mathcal{K} \subseteq \mathcal{K}'$
3. $C^{\mathcal{I}} = C^{\mathcal{I}'}$ *for all concepts* C *in* $\mathcal{K}$
4. $R^{\mathcal{I}} = R^{\mathcal{I}'}$ *for all roles* R *in* $\mathcal{K}$

This notion of an interpretation's extension is required since the rewriting creates axioms, added to a KB $\mathcal{K}'$, so we will be working with two KBs and we need to have two interpretations that agree on their common part. For the remaining of this section when we refer to π, it is to be a *mapping* from $Var(r) \rightarrow \Delta^{\mathcal{I}}$ where r is a rule. We will abstract the fact we are dealing with rules and consider only the body of the rules. Since we are not modifying the head and folding the graph around the variables referenced in the head, we can do so w.l.o.g. In order to prove the correctness of the rewriting, we will show that the steps composing the folding of a graph are valid and then show that the insertion of a rule maintains its semantics.

Lemma 1 (Label removal). *Let $\mathcal{I}$ be a model of $\mathcal{K}$, b be the body of a rule and π be a mapping.*

1. *If* $\mathcal{I} \models^{\pi} b$, *then there is a label removal* b_{lr} *of* b *and an interpretation* $\mathcal{I}'$, *extension of* $\mathcal{I}$, *such that* $\mathcal{I}' \models^{\pi} b_{lr}$
2. *If* $\mathcal{I} \models^{\pi} b_{lr}$ *for a label removal* b_{lr} *of* b, *then* $\mathcal{I} \models^{\pi} b$.

For the proof, we will define b as $C(x)$. Following the label removal we described, $\mathcal{K}'$ would be defined as $\mathcal{K} \cup \{C \sqsubseteq instC.Self\}$ and b_{lr} as $instC(x, x)$. To define $\mathcal{I}'$, we extend $\mathcal{I}$ with $(instC)^{\mathcal{I}'} = \{(a, a) \in \Delta^{\mathcal{I}'} \times \Delta^{\mathcal{I}'} \mid a \in C^{\mathcal{I}}\}$.

Proof (of (1)). Since $\mathcal{K}'$ is only augmented by a RIA with a fresh role name on the right hand side and $\mathcal{I}' \models^{\pi} \mathcal{K}$ (by definition of an extension), it is trivial to see that $\mathcal{I}' \models^{\pi} \mathcal{K}'$. To show that $\mathcal{I}' \models^{\pi} b_{lr}$, we will prove by contradiction so we assume that $\mathcal{I}' \not\models^{\pi} b_{lr}$, i.e., $\mathcal{I}' \not\models^{\pi} instC(x,x)$. We know that $\mathcal{I}' \models^{\pi} b$ so $\pi(x) \in C^{\mathcal{I}'}$. The axiom added in $\mathcal{K}'$ gives that $(\pi(x), \pi(x)) \in instC^{\mathcal{I}'}$. The semantics of b_{lr} being $(z,z) \in instC^{\mathcal{I}'}$, we just have to consider $z = \pi(x)$ to have a contradiction. □

Proof (of (2)). Since $\mathcal{K} \subseteq \mathcal{K}'$ and $\mathcal{I} \models \mathcal{K}'$, it trivially models $\mathcal{K}$. We will prove by contradiction for $\mathcal{I} \models^{\pi} b$, so let's assume that $\mathcal{I} \not\models^{\pi} b$, i.e. $\mathcal{I} \not\models^{\pi} C(x)$. Since $\mathcal{I} \models^{\pi} b_{lr}$, we have $(\pi(x), \pi(x)) \in instC^{\mathcal{I}}$. Since $instC$ is a fresh role name, it can only be introduced by the RIA which means that $\pi(x) \in C^{\mathcal{I}}$, which gives us a contradiction. □

Lemma 2 (Rolling-up). *Let $\mathcal{I}$ be a model of $\mathcal{K}$, b be the body of a rule and π be a mapping.*

1. *If $\mathcal{I} \models^{\pi} b$, then there is a rolling-up b_{ru} of b and an interpretation $\mathcal{I}'$,extension of $\mathcal{I}$, such that $\mathcal{I}' \models^{\pi} b_{ru}$*
2. *If $\mathcal{I} \models^{\pi} b_{ru}$ for a rolling-up b_{ru} of b, then $\mathcal{I} \models^{\pi} b$.*

Given that this technique is widely known and used, we will not give a proof for it here. The reader may refer to [6] for further information.

By using Lemmas 1 and 2, we can formulate the following lemma which shows that the folding of a rule graph is semantically equivalent to the original.

Lemma 3. *Let b be the body of a rule. The folding b_f of b, obtained by any composition of the rolling-up and label removal, is equivalent to b.*

Proof. The operations used in the folding of a rule graph have been shown to yield equivalent graphs. The composition of any of these operations obviously also yields equivalent output. This can be shown as follows. Let f be a function that yields a result equivalent to its output, i.e. $f(x) \equiv x$. It trivial to show that $f(f(x)) \equiv x$ since we can replace $f(x)$ by x since they are equivalent. □

Theorem 1. *Let r be a rule of the form $b_r \rightarrow h_r$, where b_r is the body and h_r is the head, and a be the axiom rewriting if this rule of the form $a_b \sqsubseteq a_h$. We then have that an interpretation $\mathcal{I}$ that models $\mathcal{K}$ and satisfies r iff there exists an interpretation $\mathcal{I}'$, extension of $\mathcal{I}$, that models $\mathcal{K}'$ and satisfies a. $\mathcal{K}'$ is defined as $\mathcal{K}$ augmented with the axioms generated by the folding and a.*

Proof. By Lemma 3, we know that b_r is equivalent to a_b. It is trivial to see that the heads are equivalent. For the "if" direction, we have $\mathcal{I} \models r$ and since $\mathcal{I}'$ is an extension of $\mathcal{I}$, it follows that $\mathcal{I}' \models r$. That means that $\mathcal{I}' \models b_r$, b_r being equivalent to a_b, we also have $\mathcal{I}' \models a_b$. Since $\mathcal{I}' \models \mathcal{K}'$, which contains $a_b \sqsubseteq a_h$, we have that $\mathcal{I}' \models a$.

For the "only if" direction, we have $\mathcal{I}' \models a$, thus $\mathcal{I}' \models a_b$ and by Lemma 3 $\mathcal{I}' \models b_r$. By the definition of an extension and the fact b_r only references roles and concepts in $\mathcal{K}$, it follows that $\mathcal{I} \models b_r$. The same applies to the head so $\mathcal{I} \models h_r$. That shows that $\mathcal{I} \models b_r$ implies $\mathcal{I} \models h_r$, thus $\mathcal{I} \models r$. □

6 Conclusion

We have presented a technique to rewrite a certain form of tree-shaped rules into DL axioms. This technique does not add expressive power to $\mathcal{SROIQ}$, but it facilitates the use of existing constructs in a non-trivial way to model knowledge intuitively expressed as rules. It also offers some advantages over previous approaches. It lets us drop the restriction to DL-safety of the rules. It is compatible with existing tools and reasoners. Since rules are embedded in the KB, we avoid using another formalism to save them. These two characteristics combined gives the most interesting feature of the technique, when compared to previously proposed technique: it is readily usable in practice. We implemented the rewriting and we have tested it. The results show that the cost of reasoning over a KB containing rewritten rules is reasonable. That is, the reasoners fare well with the combination of RIA and self-restriction we use. Furthermore, it also scales very well, both with regards to the size of the KB and the number of rules. In order to make the most of this practical usability, we are currently developing a plug-in[3][7] for Protégé[4] that enables the creation, either in textual or graphical mode, and the insertion of rules in KBs as axioms or as SWRL rules if they are not admissible, thus maximizing expressivity of the KB. In future work, we intend to investigate how $\mathcal{SROIQ}$ and/or the rewriting could be extended to handle a wider range of rules.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
2. Tsarkov, D., Riazanov, A., Bechhofer, S., Horrocks, I.: Using Vampire to reason with OWL. In McIlraith, S.A., Plexousakis, D., van Harmelen, F., eds.: Proc. of the 3rd International Semantic Web Conference (ISWC 2004). Number 3298 in Lecture Notes in Computer Science, Springer (2004) 471–485
3. Rosati, R.: DL+log: Tight integration of description logics and disjunctive datalog. In: Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2006). (2006) 68–78
4. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A semantic web rule language combining OWL and RuleML. W3C Member Submission (21 May 2004) Available at `http://www.w3.org/Submission/SWRL/`.
5. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006), AAAI Press (2006) 57–67
6. Tessaris, S.: Questions and answers: reasoning and querying in Description Logic. PhD thesis, University of Manchester (2001)
7. Gasse, F., Haarslev, V.: Dlrule: A rule editor plug-in for protégé. Proceedings of the OWLED 2008 Workshop on OWL: Experiences and Directions (2008)

[3] `http://users.encs.concordia.ca/~f_gasse/`
[4] `http://protege.stanford.edu`