

Description Logic Rules

Markus Krötzsch and Sebastian Rudolph and Pascal Hitzler¹

Abstract. We introduce *description logic (DL) rules* as a new rule-based formalism for knowledge representation in DLs. As a fragment of the Semantic Web Rule Language SWRL, DL rules allow for a tight integration with DL knowledge bases. In contrast to SWRL, however, the combination of DL rules with expressive description logics remains decidable, and we show that the DL *SROIQ* – the basis for the ongoing standardisation of OWL 2 – can completely internalise DL rules. On the other hand, DL rules capture many expressive features of *SROIQ* that are not available in simpler DLs yet. While reasoning in *SROIQ* is highly intractable, it turns out that DL rules can be introduced to various lightweight DLs without increasing their worst-case complexity. In particular, DL rules enable us to significantly extend the tractable DLs \mathcal{EL}^{++} and DLP.

1 INTRODUCTION

The development of description logics (DLs) has been driven by the desire to push the expressivity bounds of these knowledge representation formalisms while still maintaining decidability and implementability. This has led to very expressive DLs such as *SHOIN*, the logic underlying the Web Ontology Language OWL DL, *SHOIQ*, and more recently *SROIQ* [6] which is the basis for the ongoing standardisation of OWL 2² as the next version of the Web Ontology Language. On the other hand, more lightweight DLs for which most common reasoning problems can be implemented in (sub)polynomial time have also been sought, leading, e.g., to the tractable DL \mathcal{EL}^{++} [1].

Another popular paradigm of knowledge representation are rule-based formalisms – ranging from logic programming to deductive databases. Similar to DLs, the expressivity and complexity of rule languages has been studied extensively [3], and many decidable and tractable formalisms are known. Yet, reconciling DLs and rule languages is not easy, and many works have investigated this problem.

In this paper, we introduce *DL rules* as an expressive new rule language for combining DLs with first-order rules in a rather natural way that admits tight integration with existing DL systems. Since DLs can be considered as fragments of function-free first-order logic with equality, an obvious approach is to combine them with first-order Horn-logic rules. This is the basis of the *Semantic Web Rule Language SWRL* [7], proposed as a rule extension to OWL. However, reasoning becomes undecidable for the combination of OWL and SWRL, and thus more restricted rule languages have been investigated. A prominent example are *DL-safe rules* [12], which restrict the applicability of rules to a finite set of named individuals to retain decidability. Similar safety conditions have already been proposed for CARIN [11] in the context of the DL $\mathcal{ALCN}\mathcal{R}$, where also

acyclicity of rules and Tboxes was studied as an alternative for retaining decidability. Another basic approach is to identify the Horn-logic rules directly expressible in OWL DL (i.e. *SHOIN*), and this fragment has been called *Description Logic Programs DLP* [5].

DL rules in turn can be characterised as a decidable fragment of SWRL, which corresponds to a large class of SWRL rules indirectly expressible in *SROIQ*. They are based on the observation that DLs can express only tree-like interdependencies of variables. For example, the concept expression $\exists \text{worksAt.University} \sqcap \exists \text{supervises.PhDStudent}$ that describes all people working at a university and supervising some PhD student corresponds to the following first-order formula:

$$\exists y. \exists z. \text{worksAt}(x, y) \wedge \text{University}(y) \wedge \text{supervises}(x, z) \wedge \text{PhDStudent}(z)$$

Here variables form the nodes of a tree with root x , where edges are given by binary predicates. Intuitively, DL rules are exactly those SWRL rules, where premises (rule bodies) consist of one or more of such tree-shaped structures. One could, for example, formulate the following rule:

$$\text{worksAt}(x, y) \wedge \text{University}(y) \wedge \text{supervises}(x, z) \wedge \text{PhDStudent}(z) \rightarrow \text{profOf}(x, z)$$

Since SWRL allows the use of DL concept expressions in rules, we obtain *SROIQ* rules, \mathcal{EL}^{++} rules, or DLP rules as extensions of the respective DLs. For the case of *SROIQ*, DL rules have independently been proposed in [4], where a tool for editing such rules was presented. As shown below, DL rules are indeed “syntactic sugar” in this case, even though rule-based presentations are often significantly simpler due to the fact that many rules require the introduction of auxiliary vocabulary for being encoded in *SROIQ*. On the other hand, we also consider the light-weight DLs \mathcal{EL}^{++} and DLP for which DL rules truly extend expressivity, and we show that the polynomial complexity of these DLs is preserved by this extension.

After giving some notation in Section 2, we introduce DL rules in Section 3. Section 4 shows how DL rules can be internalised in *SROIQ*, while Section 5 employs a novel reasoning algorithm to process \mathcal{EL}^{++} rules. Section 6 introduces DLP 2 and shows the tractability of this DL-based rule language. Most proofs are omitted and can be found in [9].

2 PRELIMINARIES

In this section, we briefly introduce our notation based on the DL *SROIQ* [6]. More detailed definitions and introductory remarks can be found in [9]. As usual, the DLs considered in this paper are based on three disjoint sets of *individual names* N_I , *concept names* N_C , and *role names* N_R containing the *universal role* $U \in N_R$.

Definition 1 A *SROIQ Rbox* for N_R is based on a set \mathbf{R} of roles defined as $\mathbf{R} := N_R \cup \{R^- \mid R \in N_R\}$, where we set $\text{Inv}(R) := R^-$

¹ Universität Karlsruhe (TH), Germany, [mak|sru|phi]@aifb.uni-karlsruhe.de

² OWL 2 is the forthcoming W3C recommendation updating OWL, based on the OWL 1.1 member submission, cf. <http://www.w3.org/2007/OWL>.

and $\text{Inv}(R^-) := R$ to simplify notation. In the sequel, we will use the symbols R, S , possibly with subscripts, to denote roles.

A generalised role inclusion axiom (RIA) is a statement of the form $S_1 \circ \dots \circ S_n \sqsubseteq R$, and a set of such RIAs is a *SROIQ Rbox*. An Rbox is regular if there is a strict partial order $<$ on \mathbf{R} such that

- $S < R$ iff $\text{Inv}(S) < R$, and
- every RIA is of one of the forms: $R \circ R \sqsubseteq R$, $R^- \sqsubseteq R$, $S_1 \circ \dots \circ S_n \sqsubseteq R$, $R \circ S_1 \circ \dots \circ S_n \sqsubseteq R$, or $S_1 \circ \dots \circ S_n \circ R \sqsubseteq R$ such that $R \in \mathbf{N}_R$ is a (non-inverse) role name, and $S_i < R$ for $i = 1, \dots, n$.

The set of simple roles for some Rbox is defined inductively:

- If a role R occurs only on the right-hand-side of RIAs of the form $S \sqsubseteq R$ such that S is simple, then R is also simple.
- The inverse of a simple role is simple.

Definition 2 Given a *SROIQ Rbox* \mathcal{R} , the set of concept expressions \mathbf{C} is defined as follows:

- $\mathbf{N}_C \subseteq \mathbf{C}$, $\top \in \mathbf{C}$, $\perp \in \mathbf{C}$,
- if $C, D \in \mathbf{C}$, $R \in \mathbf{R}$, $S \in \mathbf{R}$ a simple role, $a \in \mathbf{N}_I$, and n a non-negative integer, then $\neg C$, $C \sqcap D$, $C \sqcup D$, $\{a\}$, $\forall R.C$, $\exists R.C$, $\exists S.\text{Self}$, $\leq n S.C$, and $\geq n S.C$ are also concept expressions.

Throughout this paper, the symbols C, D will be used to denote concept expressions. A *SROIQ Tbox* is a set of general concept inclusion axioms (GCI) of the form $C \sqsubseteq D$.

An individual assertion can have any of the following forms: $C(a)$, $R(a, b)$, $\neg R(a, b)$, $a \neq b$, with $a, b \in \mathbf{N}_I$ individual names, $C \in \mathbf{C}$ a concept expression, and $R, S \in \mathbf{R}$ roles with S simple. A *SROIQ Abox* is a set of individual assertions.

A *SROIQ* knowledge base KB is the union of a regular Rbox \mathcal{R} , and an Abox \mathcal{A} and Tbox \mathcal{T} for \mathcal{R} .

The standard semantics of the above constructs is recalled in [9].

3 DESCRIPTION LOGIC RULES

We introduce *DL rules* as a syntactic fragment of first-order logic.

Definition 3 Consider some description logic \mathcal{L} with concept expressions \mathbf{C} , individual names \mathbf{N}_I , roles \mathbf{R} (possibly including inverse roles), and let \mathbf{V} be a countable set of first-order variables. Given terms $t, u \in \mathbf{N}_I \cup \mathbf{V}$, a concept atom (role atom) is a formula of the form $C(t)$ ($R(t, u)$) with $C \in \mathbf{C}$ ($R \in \mathbf{R}$).

To simplify notation, we often use finite sets S of (role and concept) atoms for representing the conjunction $\bigwedge S$. Given such a set S of atoms and terms $t, u \in \mathbf{N}_I \cup \mathbf{V}$, a path from t to u in S is a non-empty sequence $R_1(x_1, x_2), \dots, R_n(x_n, x_{n+1}) \in S$ where $x_1 = t$, $x_i \in \mathbf{V}$ for $2 \leq i \leq n$, $x_{n+1} = u$, and $x_i \neq x_{i+1}$ for $1 \leq i \leq n$. A term t in S is initial (resp. final) if there is no path to t (resp. no path starting at t).

Given sets B and H of atoms, and a set $\mathbf{x} \subseteq \mathbf{V}$ of all variables in $B \cup H$, a description logic rule (DL rule) is a formula $\forall \mathbf{x}. \bigwedge B \rightarrow \bigwedge H$ such that

- (R1) for any $u \in \mathbf{N}_I \cup \mathbf{V}$ that is not initial in B , there is a path from exactly one initial $t \in \mathbf{N}_I \cup \mathbf{V}$ to u in B ,
- (R2) for any $t, u \in \mathbf{N}_I \cup \mathbf{V}$, there is at most one path in B from t to u ,
- (R3) if H contains an atom $C(t)$ or $R(t, u)$, then t is initial in B .

Here $\forall \mathbf{x}$ for $\mathbf{x} = \{x_1, \dots, x_n\}$ abbreviates an arbitrary sequence $\forall x_1 \dots \forall x_n$. Since we consider only conjunctions with all variables quantified, we will often simply write $B \rightarrow H$ instead of $\forall \mathbf{x}. \bigwedge B \rightarrow \bigwedge H$. A rule base RB for some DL \mathcal{L} is a set of DL rules for \mathcal{L} .

The semantics of DL rules in the context of a description logic knowledge base is given by interpreting both the rules and knowledge base as first-order theories in the usual way, and applying the standard semantics of predicate logic. This has been discussed in the context of SWRL in [7], and we will not repeat the details here.

Definition 3 ensures that role atoms in rule bodies essentially form a (set of) directed trees, starting at initial elements. Using the well-known equivalence of formulae $\{p \rightarrow q_1 \wedge q_2\}$ and $\{p \rightarrow q_1, p \rightarrow q_2\}$, one can transform any rule into an equivalent set of rules without conjunctions in rule heads. This can be done in linear time, so we assume without loss of generality that all DL rules are of this form.

Since all DLs considered herein support nominals, we can also assume that all terms in rules are variables. Indeed, any atom $C(a)$ with $a \in \mathbf{N}_I$ can be replaced by $C(x) \wedge \{a\}(x)$ with $x \in \mathbf{V}$ a new variable. Using inverse roles, role atoms with individual names can be replaced by concept atoms as follows: $R(x, a)$ becomes $\exists R.\{a\}(x)$, $R(a, y)$ becomes $\exists \text{Inv}(R).\{a\}(y)$, and $R(a, b)$ becomes $\exists R.\{b\}(x) \wedge \{a\}(x)$. A similar transformation is possible for rule heads, where generated concept atoms $\{a\}(x)$ are again added to the rule body.

Before considering the treatment of DL rules in concrete DLs, we highlight some relevant special applications of DL rules.

Concept products Rules of the form $C(x) \wedge D(y) \rightarrow R(x, y)$ can encode *concept products* (sometimes written $C \times D \sqsubseteq R$) asserting that all elements of two classes must be related [13]. Examples include statements such as $\text{Elephant}(x) \wedge \text{Mouse}(y) \rightarrow \text{biggerThan}(x, y)$ or $\text{Alkaline}(x) \wedge \text{Acid}(y) \rightarrow \text{neutralises}(x, y)$.

Local reflexivity, universal role Rules of the forms $C(x) \rightarrow R(x, x)$ and $R(x, x) \rightarrow C(x)$ can replace the *SROIQ* Tbox expression $C \sqsubseteq \exists R.\text{Self}$ and $\exists R.\text{Self} \sqsubseteq C$. The universal role U of *SROIQ* can be defined as $\top(x) \wedge \top(y) \rightarrow U(x, y)$. Hence, a DL that permits such rules does not need to explicitly introduce those constructs.

Qualified RIAs DL rules of course can express arbitrary role inclusion axioms, but they also can state that a RIA applies only to instances of certain classes. Examples include $\text{Woman}(x) \wedge \text{hasChild}(x, y) \rightarrow \text{motherOf}(x, y)$ and $\text{trusts}(x, y) \wedge \text{Doctor}(y) \wedge \text{recommends}(y, z) \wedge \text{Medicine}(z) \rightarrow \text{buys}(x, z)$.

4 DL RULES IN SROIQ

We now show how knowledge bases of such rules can be internalised into the DL *SROIQ*. Since *SROIQ* supports inverse roles, it turns out that one can relax condition (R1) of DL rules as follows:

- (R1') for any $u \in \mathbf{N}_I \cup \mathbf{V}$ that is not initial in B , there is a path from one or more initial elements $t \in \mathbf{N}_I \cup \mathbf{V}$ to u in B .

On the other hand, we need to adopt the notions of *regularity* and *simplicity* to DL rule bases in *SROIQ*, which again restricts the permissible rule bases:

Definition 4 Consider a rule base RB and a knowledge base KB for *SROIQ*. The set of simple roles of $\text{KB} \cup \text{RB}$ is the smallest set of roles containing every role R for which the following hold:

- If R or $\text{Inv}(R)$ occur on the right-hand-side of some RIA of KB , then this RIA is of the form $S \sqsubseteq R$ or $S \sqsubseteq \text{Inv}(R)$, and S is simple.
- If R or $\text{Inv}(R)$ occur in some rule head of the form $R(x, y)$ or $\text{Inv}(R)(x, y)$ in RB , then the according rule body is of the form $S(x, y)$ with S simple, or of the form $C(x)$ where $x = y$.

Note that this is indeed a proper inductive definition, where roles that do not occur on the right of either RIAs or rules form the base case.

The extended knowledge base $\text{KB} \cup \text{RB}$ is admissible for *SROIQ* if all roles $S_{(i)}$ occurring in concept (sub)expressions of the form $\leq n S.C$, $\geq n S.C$, $\exists S.\text{Self}$, and $\text{Dis}(S_1, S_2)$, and in role atoms of the form $S(x, x)$ ($x \in \mathbf{V}$) are simple.

An extended knowledge base $\text{KB} \cup \text{RB}$ is regular if there is a strict partial order $<$ on \mathbf{R} such that

- $S < R$ iff $\text{Inv}(S) < R$,
- the role box of KB is regular w.r.t. $<$, and
- for any rule $B \rightarrow R(x, y)$, each $S(z, v) \in B$ satisfies one of the following:
 - $S < R$, or
 - there is no path from v to y , or
 - $S = R$, there is no other $R(z', v') \in B$ with a path from v' to y , and we find that: either $x = z$ and there is no $C(x) \in B$, or $y = v$ and there is no $C(y) \in B$.

Note that RIAs in regular *SROIQ* knowledge bases are allowed to have two special forms for transitivity and symmetry, which we do omit for the definition of regularity in DL rules to simplify notation. Since S in $S(x, x)$ is simple, we can replace such role atoms by concept atoms $C(x)$ where C is a new concept name for which a new axiom $C \equiv \exists S.\text{Self}$ is added. We will thus assume that no role atoms of this form occur in admissible knowledge bases.

One can now show that checking the satisfiability of extended *SROIQ* knowledge bases that are admissible and regular is decidable, and has the same worst-case complexity as reasoning in *SROIQ*. This is achieved by a polynomial transformation of rule bases into *SROIQ* axioms. The first step of doing this is to replace “dead branches” of the tree-shaped query body by DL concepts. The proof is a variation of the “rolling-up” technique used for conjunctive query answering [2].

Lemma 5 Any DL rule $B \rightarrow H$ for *SROIQ* can be transformed into a semantically equivalent rule $B' \rightarrow H$ such that all paths in B' are contained in a single maximal path. If $H = R(x, y)$, then y is the final element of that maximal path, and if $H = C(x)$ then there are no paths in B . A rule with these properties is called linearised.

As an example, the DL rule that was given in the introduction can be simplified to yield (using “,” instead of “ \wedge ” for brevity):

$\exists \text{worksAt.University}(x), \text{supervises}(x, z), \text{PhDStudent}(z) \rightarrow \text{profOf}(x, z)$

The above transformation allows us to reduce tree-shaped rules to rules of only linear structure that are much more similar to RIAs in *SROIQ*. But while all role atoms now belong to a single maximal path, rules might still contain disconnected concept atoms. The rule $R(x, y) \wedge S(u, v) \wedge C(z) \rightarrow T(x, v)$, e.g., is rewritten to $\exists R.\tau(x) \wedge S(u, v) \wedge C(z) \rightarrow T(x, v)$.

Now it can be shown that DL rules in *SROIQ* can be internalised.

Theorem 6 Consider a rule base RB and a knowledge base KB for *SROIQ*, such that $\text{RB} \cup \text{KB}$ is admissible. There is a *SROIQ* knowledge base KB_{RB} that can be computed in time polynomial in the size of RB , such that $\text{KB} \cup \text{RB}$ and $\text{KB} \cup \text{KB}_{\text{RB}}$ are equisatisfiable.

Moreover, if $\text{KB} \cup \text{RB}$ is regular, then $\text{KB} \cup \text{KB}_{\text{RB}}$ is also regular.

Proof. We can assume rules in RB to have the form as in Lemma 5, since the transformation given in [9] preserves regularity and simplicity in $\text{KB} \cup \text{RB}$. We assume w.l.o.g. that no rule in RB has the universal role U in its head: such rules would be tautological. We

further assume that all variables occurring in rule heads also occur in their body – atoms of the form $\top(x)$ can safely be added to that end.

For any rule $B \rightarrow R(x, y)$, Lemma 5 asserts that B contains at most one maximal path with final element y , and all role atoms of B (if any) are part of that path. Let z be the initial element of this path if it exists, and let z be y otherwise. If $x \neq z$, then x occurs in B only in concept atoms $C(x)$, and we can add a role atom $U(x, z)$ to B without violating (R1)–(R3). This change preserves the semantics of the rule since $U(x, z)$ is true for any variable assignment (mapping free variables to domain elements of \mathcal{I} ; sometimes also called *variable binding* [7]) in any interpretation. Regularity of the role base is preserved since we can assume w.l.o.g. U to be the least element of $<$ (exploiting that U does not occur in rule heads). Simplicity is no concern as R by assumption is not a simple role in $\text{KB} \cup \text{RB}$. In summary, we may transform the body of any rule with head $R(x, y)$ to contain exactly one maximal path, leading from x to y .

We describe the step-wise computation of KB_{RB} . Initially, we set $\text{KB}_{\text{RB}} := \emptyset$, and define the set of remaining rules as $\text{RB}' := \text{RB}$. The reduction proceeds iteratively until RB' is empty. In every step, we select some rule $B \rightarrow H \in \text{RB}$. As discussed above, there is only a single maximal path of roles in B , all role atoms in B are part of that path, and all but adjacent variables in the path are distinct (no cycles). We distinguish five cases:

- (1) If B contains atoms $D(z)$ and $D'(z)$ for some variable z , then these atoms are replaced in B by a new atom $(D \sqcap D')(z)$.
- (2) Otherwise, if $H = C(x)$ and $B = D(x)$, then $B \rightarrow H$ is removed from RB' , and a Tbox axiom $D \sqsubseteq C$ is inserted into KB_{RB} .
- (3) Otherwise, if $H = R(x, y)$ and B is of the form $\{R_1(x, x_2), \dots, R_n(x_n, y)\}$, then $B \rightarrow H$ is removed from RB' , and an Rbox axiom $R_1 \circ \dots \circ R_n \sqsubseteq R$ is inserted into KB_{RB} .
- (4) Otherwise, if $H = R(x, y)$, and there is $D(z) \in B$ such that z occurs in a role atom of B or H (in first or second argument position), then the following is done. First, a new role name S is introduced, and the Tbox axiom $D \equiv \exists S.\text{Self}$ is added to KB_{RB} . Second, a new variable $z' \in \mathbf{V}$ is introduced, the role atom $S(z, z')$ is added to B , every role atom $T(x', z) \in B$ is replaced by $T(x', z')$, and every role atom $T(z, y') \in B$ is replaced by $T(z', y')$. Finally, $D(z)$ is removed from B , and if $z = y$ then the rule head is replaced by $R(x, z')$.
- (5) Otherwise, if $H = C(x)$ or $H = R(x, y)$, and there is some $D(z) \in B$ such that z occurs neither in H nor in any role atom of B , then the following is done. If B contains some atom of the form $R(x, t)$ so there is no atom $D'(x) \in B$, then define $u := y$; otherwise define $u := x$. Now $D(z)$ in B is replaced by the atom $\exists U.D(u)$.

The correctness of this procedure can be established by verifying the following claims:

1. The cases distinguished by the algorithm are exhaustive.
2. The algorithm terminates after a polynomial number of steps.
3. After termination, $\text{KB} \cup \text{KB}_{\text{RB}}$ is a *SROIQ* knowledge base.
4. After termination, $\text{KB} \cup \text{RB}$ and $\text{KB} \cup \text{KB}_{\text{RB}}$ are equisatisfiable.
5. If $\text{KB} \cup \text{RB}$ is regular, then so is $\text{KB} \cup \text{KB}_{\text{RB}}$.

Details on the proofs of those claims can be found in [9]. \square

Considering again our introductory example, we arrive at the following *SROIQ* axioms (where S_1, S_2 are new auxiliary roles):

$$S_1 \circ \text{supervises} \circ S_2 \sqsubseteq \text{profOf} \\ \exists \text{worksAt.University} \equiv \exists S_1.\text{Self} \quad \text{PhDStudent} \equiv \exists S_2.\text{Self}$$

Based on Theorem 6, we conclude that the problem of checking the

satisfiability of *SROIQ* knowledge bases extended with DL rules is decidable, as long as the extended knowledge base is admissible and regular. Since the internalisation is possible in polynomial time, the worst-case complexity for this problem is the same as for checking satisfiability of *SROIQ* knowledge bases.

5 DL RULES IN \mathcal{EL}^{++}

In this section, we investigate DL rules for the DL \mathcal{EL}^{++} [1], for which many typical inference problems can be solved in polynomial time. As \mathcal{EL}^{++} cannot internalise DL rules, they constitute a true extension of expressivity. We therefore take a different approach than in *SROIQ*: instead of considering rule bases as an auxiliary set of axioms that is successively reduced and internalised, we introduce DL rules as core expressive mechanism to which all other \mathcal{EL}^{++} axioms can be reduced. While \mathcal{EL}^{++} rule bases offer many expressive features formerly unavailable in \mathcal{EL}^{++} , we show that the complexity of core inference problems remains tractable. We simplify our presentation by omitting concrete domains from \mathcal{EL}^{++} – they are not affected by our extension and can be treated as shown in [1].

Definition 7 A role of \mathcal{EL}^{++} is a (non-inverse) role name. An \mathcal{EL}^{++} Rbox is a set of generalised role inclusion axioms, and an \mathcal{EL}^{++} Tbox (Abox) is a *SROIQ* Tbox (Abox) that contains only the following concept constructors: \sqcap , \exists , \top , \perp , as well as nominal classes $\{a\}$. An \mathcal{EL}^{++} knowledge base is the union of an \mathcal{EL}^{++} Rbox, Tbox and Abox. An \mathcal{EL}^{++} rule base is a set of DL rules for \mathcal{EL}^{++} that do not contain atoms of the form $R(x, x)$ in the body.

Note that we do not have any requirement for regularity or simplicity of roles in the context of \mathcal{EL}^{++} . It turns out that neither is relevant for obtaining decidability or tractability. The case of $R(x, x)$ in bodies is not addressed by the below algorithm – [10] significantly extends the below approach to cover this and other features. Since it is obvious that both concept and role inclusion axioms can directly be expressed by DL rules, we will consider only \mathcal{EL}^{++} rule bases without any additional \mathcal{EL}^{++} knowledge base axioms. We can restrict our attention to \mathcal{EL}^{++} rules in a certain normal form:

Definition 8 An \mathcal{EL}^{++} rule base RB is in normal form if all concept atoms in rule bodies are either concept names or nominals, all variables in a rule’s head also occur in its body, and all rule heads are of one of the following forms:

$$A(x) \quad \exists R.A(x) \quad R(x, y)$$

where $A \in \mathbf{N}_C \cup \{\{a\} \mid a \in \mathbf{N}_I\} \cup \{\top, \perp\}$ and $R \in \mathbf{N}_R$. A set \mathcal{B} of basic concept expressions for RB is defined as $\mathcal{B} := \{C \mid C \in \mathbf{N}_C, C \text{ occurs in RB}\} \cup \{\{a\} \mid a \in \mathbf{N}_I, a \text{ occurs in RB}\} \cup \{\top, \perp\}$.

Proposition 9 Any \mathcal{EL}^{++} rule base can be transformed into an equisatisfiable \mathcal{EL}^{++} rule base in normal form. The transformation can be done in polynomial time.

When checking satisfiability of \mathcal{EL}^{++} rule bases, we can thus restrict to rule bases in the above normal form. A polynomial algorithm for checking class subsumptions in \mathcal{EL}^{++} knowledge bases has been given in [1], and it was shown that other standard inference problems can easily be reduced to that problem. We now present a new algorithm for checking satisfiability of \mathcal{EL}^{++} rule bases, and show its correctness and tractability. Clearly, subsumption checking can be reduced to this problem: given a new individual $a \in \mathbf{N}_I$, the rule base $\text{RB} \cup \{C(a), \{a\}(x) \sqcap D(x) \rightarrow \perp(x)\}$ is unsatisfiable iff RB entails $C \sqsubseteq D$. Instance checking in turn is directly reducible to subsumption checking in the presence of nominals.

Algorithm 10 The algorithm proceeds by computing two sets: a set \mathcal{E} of inferred “domain elements”, and a set \mathcal{S} of relevant subclass inclusion axioms that are entailed by RB. The elements of \mathcal{E} are represented by basic concept expressions of RB, i.e. $\mathcal{E} \subseteq \mathcal{B}$, and the inclusion axioms in \mathcal{S} are of the form $C \sqsubseteq D$ or $C \sqsubseteq \exists R.D$, where $C, D \in \mathcal{E}$. Thus \mathcal{E} and \mathcal{S} are polynomially bounded by the size of RB.

Initially, we set $\mathcal{E} := \{\{a\} \mid \{a\} \in \mathcal{B}\} \cup \{\top\}$ and $\mathcal{S} := \emptyset$. Now a DL rule is applied whenever we find that there is a match with the rule body. Given a rule $B \rightarrow H$, a match θ is a mapping from all variables in B to elements of \mathcal{E} , such that the following hold:

- for every $C(y) \in B$, $\theta(y) \sqsubseteq C \in \mathcal{S}$, and
- for every $R(y, z) \in B$, $\theta(y) \sqsubseteq \exists R.\theta(z) \in \mathcal{S}$.

The algorithm now proceeds by applying the following rules until no possible rule application further modifies the set \mathcal{E} or \mathcal{S} :

- (EL1) If $C \in \mathcal{E}$, then $\mathcal{S} := \mathcal{S} \cup \{C \sqsubseteq C, C \sqsubseteq \top\}$.
- (EL2) If there is a rule $B \rightarrow E(x) \in \text{RB}$, and if there is a match θ for B with $\theta(x) = \theta_x$, then $\mathcal{S} := \mathcal{S} \cup \{\theta_x \sqsubseteq E\}$. In this case, if $E = C$ or $E = \exists R.C$, then $\mathcal{E} := \mathcal{E} \cup \{C\}$.
- (EL3) If there is a rule $B \rightarrow R(x, y) \in \text{RB}$, and if there is a match θ for B with $\theta(x) = \theta_x$ and $\theta(y) = \theta_y$, then $\mathcal{S} := \mathcal{S} \cup \{\theta_x \sqsubseteq \exists R.\theta_y\}$.
- (EL4) If $\{C \sqsubseteq \{a\}, D \sqsubseteq \{a\}, D \sqsubseteq E\} \subseteq \mathcal{S}$ then $\mathcal{S} := \mathcal{S} \cup \{C \sqsubseteq E\}$.

Here we assume that $C, D, D' \in \mathcal{B}$, $E \in \mathcal{B} \cup \{\exists R.C \mid C \in \mathcal{B}\}$, and $R \in \mathbf{N}_R$. After termination, the algorithm returns “unsatisfiable” if $\perp \in \mathcal{E}$, and “satisfiable” otherwise.

The correctness of the above algorithm is shown by using the sets \mathcal{E} and \mathcal{S} for constructing a model, which is indeed possible whenever $\perp \notin \mathcal{E}$ (see [9] for details). \mathcal{EL}^{++} has a small model property (a consequence of the proofs for [1]) that allows us to consider at most one individual for representing the members of each class. The set \mathcal{E} thus is used to record classes which must have some element, and matches θ use these class names to represent (arbitrary) individuals to which some DL rule might be applied.

Assuming that all steps of Algorithm 10 are computable in polynomial time, it is easy to see that the algorithm also terminates in polynomial time, since there are only polynomially many possible elements for \mathcal{E} and \mathcal{S} , and each case adds new elements to either set. However, it also has to be verified that individual steps can be computed efficiently, and this is not obvious for the match-checks in (EL2) and (EL3). Indeed, finding matches in query graphs is known to be NP-complete in general, and the tree-like structure of queries is crucial to retain tractability. Moreover, even tree-like rule bodies admit exponentially many matches. But note that Algorithm 10 does not consider *all matches* but only the (polynomially many) possible values of θ_x (and θ_y). It turns out that there is indeed an algorithm that checks in polynomial time whether a match θ as in (EL2) and (EL3) exists, but without explicitly considering all possible matches. This task is closely related to the problem of testing the existence of homomorphisms between trees and graphs.

Proposition 11 Consider a rule of the form $B \rightarrow C(x)$ ($B \rightarrow R(x, y)$), sets \mathcal{E} and \mathcal{S} as in Algorithm 10, and an element $\theta_x \in \mathcal{E}$ (elements $\theta_x, \theta_y \in \mathcal{E}$). There is an algorithm that decides whether there is a match θ such that $\theta(x) = \theta_x$ ($\theta(x) = \theta_x$ and $\theta(y) = \theta_y$), running in polynomial time w.r.t. the size of the inputs.

Theorem 12 Algorithm 10 is a sound and complete procedure for checking satisfiability of \mathcal{EL}^{++} rule bases. Satisfiability checking, instance retrieval, and computing class subsumptions for \mathcal{EL}^{++} rule bases is possible in polynomial time in the size of the rule base.

6 DLP 2

Description Logic Programs (DLP) have been proposed as a tractable formalism for bridging the gap between DL and (Horn) logic programming [5]. DLP can naturally be extended with DL rules and various other features of *SROIQ*, and the resulting tractable rule language might be dubbed DLP 2 in analogy to the ongoing standardisation of OWL 2.

A detailed syntactic characterisation for DLP was given in [14], and a yet more general formulation can be obtained from [8]. Here, we adopt a much simpler definition that focusses on the essential expressive features only:

Definition 13 *Roles of DLP are defined as in SROIQ, including inverse roles. A DLP body concept is any SROIQ concept expression that includes only concept names, nominals, \sqcap , \exists , \top , and \perp . A DLP head concept is any SROIQ concept expression that includes only concept names, nominals, \sqcap , \forall , \top , \perp , and expressions of the form $\leq 1.C$ where C is a DLP body concept.*

A DLP knowledge base is a set of *Rbox* axioms of the form $R \sqsubseteq S$ and $R \circ R \sqsubseteq R$, *Tbox* axioms of the form $C \sqsubseteq D$, and *Abox* axioms of the form $D(a)$ and $R(a, b)$, where $C \in \mathbf{C}$ is a body concept, $D \in \mathbf{C}$ is a head concept, and $a, b \in \mathbf{N}_I$ are individual names. A DLP rule base is a set of DL rules such that all concepts in rule bodies are body concepts, and all concepts in rule heads are head concepts.

A DLP 2 knowledge base consists of a DLP knowledge base that additionally might contain *Rbox* axioms of the form $\text{Dis}(R, S)$ and $\text{Asy}(R)$, together with some DLP rule base.

Dis and Asy assert role disjointness and asymmetry as explained in [9]. Note that neither regularity nor simplicity restrictions apply in DLP. Combining “rolling-up” as in Lemma 5 with a decomposition of the remaining single paths to multiple rules, any DLP 2 knowledge base can be transformed into an equisatisfiable set of function-free first-order Horn rules with at most five variables per formula, and this transformation is possible in polynomial time. Since this fragment of Horn-logic is tractable, we can conclude the following:

Theorem 14 *Satisfiability checking, instance retrieval, and computing class subsumptions for DLP 2 knowledge bases is possible in polynomial time in the size of the knowledge base.*

7 CONCLUSION

We have introduced *DL rules* as a rule-based formalism for augmenting description logic knowledge bases. For all DLs considered in this paper – *SROIQ*, \mathcal{EL}^{++} , and DLP – the extension with DL rules does not increase the worst-case complexity. In particular, \mathcal{EL}^{++} rules and the extended DLP 2 allow for polynomial time reasoning for common inference tasks, even though DL rules do indeed provide added expressive features in those cases.

The main contributions of this paper therefore are twofold. Firstly, we have extended the expressivity of two tractable DLs while preserving their favourable computational properties. The resulting formalisms of \mathcal{EL}^{++} rules and DLP 2 are arguably close to being maximal tractable fragments of *SROIQ*. In particular, note that the union of \mathcal{EL}^{++} and DLP is no longer tractable, even when disallowing number restrictions and inverse roles: this follows from the fact that this DL contains the DL Horn- $\mathcal{FL}\mathcal{E}$ which was shown to be ExpTime -complete in [8].

Secondly, while DL rules do not truly add expressive power to *SROIQ*, our characterisation and reduction methods for DL rules

provides a basis for developing ontology modelling tools. Indeed, even without any further extension, the upcoming OWL 2 standard would support all DL rules. Hence OWL-conformant tools can choose to provide rule-based user interfaces (as done for Protégé in [4]), and rule-based tools may offer some amount of OWL support. We remark that in the case of DLP and \mathcal{EL}^{++} , the conditions imposed on DL rules can be checked individually for each rule without considering the knowledge base as a whole. Moreover, in order to simplify rule editing, the general syntax of DL rules can be further restricted without sacrificing expressivity, e.g. by considering only chains rather than trees for rule bodies. We thus argue that DL rules can be a useful interface paradigm for many application fields.

Our treatment of rules in \mathcal{EL}^{++} and DLP 2 – used only for establishing complexity bounds in this paper – can be the basis for novel rule-based reasoning algorithms for those DLs, and we leave it for future research to explore this approach.

ACKNOWLEDGEMENTS

Research reported herein was supported by the EU in the IST projects ACTIVE (IST-2007-215040) and NeOn (IST-2006-027595), and by the German Research Foundation under the ReaSem project.

REFERENCES

- [1] Franz Baader, Sebastian Brandt, and Carsten Lutz, ‘Pushing the EL envelope’, in *Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, Edinburgh, UK, (2005). Morgan-Kaufmann Publishers.
- [2] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini, ‘On the decidability of query containment under constraints’, in *Proc. 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS’98)*, pp. 149–158. ACM Press, (1998).
- [3] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov, ‘Complexity and expressive power of logic programming’, *ACM Computing Surveys*, **33**, 374–425, (2001).
- [4] Francis Gasse, Ulrike Sattler, and Volker Haarslev. Rewriting rules into *SROIQ* axioms. Poster at 21st Int. Workshop on DLs (DL-08), 2008.
- [5] Benjamin N. Groszof, Ian Horrocks, Raphael Volz, and Stefan Decker, ‘Description logic programs: combining logic programs with description logic’, in *Proc. 12th Int. Conf. on World Wide Web (WWW 2003)*, pp. 48–57. ACM, (2003).
- [6] Ian Horrocks, Oliver Kutz, and Ulrike Sattler, ‘The even more irresistible *SROIQ*’, in *Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2006)*, pp. 57–67. AAAI Press, (2006).
- [7] Ian Horrocks and Peter F. Patel-Schneider, ‘A proposal for an OWL rules language’, in *Proc. 13th Int. Conf. on World Wide Web (WWW 2004)*, eds., Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills, pp. 723–731. ACM, (2004).
- [8] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler, ‘Complexity boundaries for Horn description logics’, in *Proc. 22nd AAAI Conf. (AAAI’07)*, (2007).
- [9] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler, ‘Description logic rules (extended technical report)’, Technical report, Universität Karlsruhe, Germany, (FEB 2008). Available at http://korrekt.org/page/SROIQ_Rules.
- [10] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler, ‘ELP: Tractable rules for OWL 2’, Technical report, Universität Karlsruhe, Germany, (May 2008). <http://korrekt.org/page/ELP>.
- [11] Alon Y. Levy and Marie-Christine Rousset, ‘Combining Horn rules and description logics in CARIN’, *Artificial Intelligence*, **104**, 165–209, (1998).
- [12] Boris Motik, Ulrike Sattler, and Rudi Studer, ‘Query answering for OWL-DL with rules’, *J. Web Sem.*, **3**(1), 41–60, (2005).
- [13] Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler, ‘All elephants are bigger than all mice’, in *Proc. 21st Int. Workshop on Description Logics (DL-08)*, (2008).
- [14] Raphael Volz, *Web Ontology Reasoning with Logic Databases*, Ph.D. dissertation, Universität Karlsruhe (TH), Germany, 2004.