ELP: Tractable Rules for OWL 2

Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler

Institut AIFB, Universität Karlsruhe, Germany

Abstract. We introduce ELP as a decidable fragment of the Semantic Web Rule Language (SWRL) that admits reasoning in polynomial time. ELP is based on the tractable description logic $\mathcal{EL}^{++}$, and encompasses an extended notion of the recently proposed *DL rules* for that logic. Thus ELP extends $\mathcal{EL}^{++}$ with a number of features introduced by the forthcoming OWL 2, such as disjoint roles, local reflexivity, certain range restrictions, and the universal role. We present a reasoning algorithm based on a translation of ELP to Datalog, and this translation also enables the seamless integration of DL-safe rules into ELP. While reasoning with DL-safe rules as such is already highly intractable, we show that DL-safe rules based on the Description Logic Programming (DLP) fragment of OWL 2 can be admitted in ELP without losing tractability.

1 Introduction

The description logic (DL) family of knowledge representation formalisms has been continuously developed for many years, leading to highly expressive (and complex), yet decidable languages. The most prominent such language is currently $\mathcal{SROIQ}$ [1], which is also the basis for the ongoing standardisation of the new *Web Ontology Language OWL 2*.[1] On the other hand, there has also been considerable interest in more light-weight languages that allow for polynomial time reasoning algorithms. DL-based formalisms that fall into that category are $\mathcal{EL}^{++}$ [2], DL Lite [3], and DLP [4]. While DL Lite strives for sub-polynomial reasoning, $\mathcal{EL}^{++}$ and DLP both are P-complete fragments of $\mathcal{SROIQ}$. In spite of this similarity, $\mathcal{EL}^{++}$ and DLP pursue different approaches towards tractability, and the combination of both is already highly intractable [5].

In this paper, we reconcile $\mathcal{EL}^{++}$ and DLP in a novel rule-based knowledge representation language ELP. While ELP can be viewed as an extension of both formalisms, however, it limits the interactions between the expressive features of either language and thus preserves polynomial time reasoning complexity. ELP also significantly extends $\mathcal{EL}^{++}$ by local reflexivity, concept products, conjunctions of simple roles, and limited range restrictions as in [6]. These features in part are already anticipated for the $\mathcal{EL}^{++}$ based language profile of OWL 2, but, to the best of our knowledge, this work is the first to establish their joint tractability.

The reasoning algorithms presented herein are based on a polynomial reduction of ELP knowledge bases to a specific kind of Datalog programs that can be evaluated in polynomial time. Since the Datalog reduction as such is comparatively simple, this

[1] OWL 2 is the forthcoming W3C recommendation for updating OWL, and is based on the OWL 1.1 member submission. See `http://www.w3.org/2007/OWL`.

outlines an interesting new implementation strategy for the $\mathcal{EL}^{++}$ profile of OWL 2: Besides the possibility of reusing optimisation methods from deductive databases, the compilation of $\mathcal{EL}^{++}$ to Datalog also provides a practical approach for extending $\mathcal{EL}^{++}$ with DL-safe rules [7]. In these respects, the presented approach bears similarities with the KAON2 transformation of $\mathcal{SHIQ}$ knowledge bases into disjunctive Datalog programs [8], though the actual algorithms are very different due to the different DLs that are addressed. DL-safe rules add new expressivity but their entailments are specifically restricted for preserving decidability – an extended example will illustrate the effects.

For this paper, we chose a presentation of ELP based on *DL rules*, a decidable subset of the Semantic Web Rule Language SWRL that has been recently proposed in two independent works [9, 10]. As shown in [9], it is possible to indirectly express such rules by means of the expressive features of $\mathcal{SROIQ}$, and large parts of ELP can still be regarded as a subset of $\mathcal{SROIQ}$. The following examples illustrate the correspondence between DLs and DL rules, and give some intuition for the expressivity of ELP:

Concept inclusions DL *Tbox* axioms $C \sqsubseteq D$ for subconcept relationships correspond to rules of the form $C(x) \rightarrow D(x)$.

Role inclusions DL *Rbox* axioms $R \circ S \sqsubseteq T$ express inclusions with role chains that correspond to rules of the form $R(x,y) \wedge S(y,z) \rightarrow T(x,z)$.

Local reflexivity The DL concept $\exists R.\mathsf{Self}$ of all things that have an R relation to themselves is described by the expression $R(x,x)$. For example, the axiom $\exists\, \mathsf{loves.Self} \sqsubseteq \mathsf{Narcist}$ corresponds to $\mathsf{loves}(x,x) \rightarrow \mathsf{Narcist}(x)$.

Role disjointness Roles in $\mathcal{SROIQ}$ can be declared disjoint to state that individuals related by one role must not be related by the other. An according example rule is $\mathsf{HasSon}(x,y) \wedge \mathsf{HasHusband}(x,y) \rightarrow \bot(x)$ ($\bot$ denoting the empty concept).

Concept products and the universal role Concept products have, e.g., been studied in [11]. The statement that all elephants are bigger than all mice corresponds to the axiom $\mathsf{Elephant} \times \mathsf{Mouse} \sqsubseteq \mathsf{biggerThan}$ and to the rule $\mathsf{Elephant}(x) \wedge \mathsf{Mouse}(y) \rightarrow \mathsf{biggerThan}(x,y)$. The universal role U that relates all pairs of individuals can be expressed by the rule $\rightarrow U(x,y)$ or as the product of the $\top$ concept with itself.

Qualified role inclusions Rules can be used to restrict role inclusions to certain concepts, which is not directly possible in $\mathcal{SROIQ}$. An example is given by the rule $\mathsf{Woman}(x) \wedge \mathsf{hasChild}(x,y) \rightarrow \mathsf{motherOf}(x,y)$.

While this work is conceptually based on [9], it significantly differs from the latter by following a completely new reasoning approach instead of extending the known algorithm for $\mathcal{EL}^{++}$. While our use of Datalog may still appear similar in spirit, the model constructions in the proofs expose additional technical complications that arise due to the novel combination of concept products, role conjunctions, and local reflexivity. Moreover, the proposed integration of DL-safe rules is not trivial since, in the absence of inverse roles, it cannot be achieved by the usual approach for "rolling-up" nested expressions, and termination of the modified transformation is less obvious.

The paper proceeds by first recalling some minimal preliminaries regarding DLs, SWRL rules, and DL-safety. Thereafter, we introduce ELP based on DL Rules for the DL $\mathcal{EL}^{++}$, and continue by giving an extended example of an ELP rule base. The next

section then presents the Datalog reduction as the basis of our reasoning algorithms, before we proceed to establish the overall reasoning complexity for ELP. We conclude the paper with a discussion of our results and some further pointers to related work. Many proofs were omitted or replaced by intuitive sketches due to space restrictions. The complete technical details can be found in the technical report [12].

2 DLs, Rules, and DL-Safety

This section gives some basic notions of description logics (DL) [13], and introduces rules that are logically similar to the Semantic Web Rule Language SWRL [14]. Such rules may include DL concept expressions, and thus generalise the common DL axiom types of Abox, Tbox, and Rbox. We thus restrict our presentation to rules, the general form of which we will later restrict to obtain favourable computational properties.

The logics considered in this paper are based on three disjoint sets of *individual names* N_I, *concept names* N_C, and *role names* N_R. Throughout this paper, we assume that these basic alphabets are finite, and consider them to be part of the given knowledge base when speaking about the "size of a knowledge base." We assume N_R to be the union of two disjoint sets of *simple roles* $\mathsf{N}_R^{\mathsf{s}}$ and *non-simple roles* $\mathsf{N}_R^{\mathsf{n}}$. Later on, the use of simple roles in conclusions of logical axioms will be restricted to ensure, intuitively speaking, that relationships of these roles are not implied by *chains* of other role relationships. In exchange, simple roles might be used in the premises of logical axioms as part of role conjunctions and reflexivity statements where non-simple roles might lead to undecidability. Fixing sets of simple and non-simple role names simplifies our presentation – in practice one could of course also check, for a given knowledge base, whether each role name satisfies the requirements for belonging to either $\mathsf{N}_R^{\mathsf{n}}$ or $\mathsf{N}_R^{\mathsf{s}}$.

Definition 1. *The set* $\mathbf{C}$ *of* concept expressions *of the DL* $\mathcal{SHOQ}$ *is defined as follows:*

- $\mathsf{N}_C \subseteq \mathbf{C}$, $\top \in \mathbf{C}$, $\bot \in \mathbf{C}$,
- *if* $C, D \in \mathbf{C}$, $R \in \mathsf{N}_R$, $S \in \mathsf{N}_R^{\mathsf{s}}$, $a \in \mathsf{N}_I$, *and* n *a non-negative integer, then* $\neg C$, $C \sqcap D$, $C \sqcup D$, $\{a\}$, $\forall R.C$, $\exists R.C$, $\leq n\, S.C$, *and* $\geq n\, S.C$ *are also concept expressions.*

The semantics of these concepts is recalled below (see also Table 1). We present $\mathcal{SHOQ}$ as a well-known DL that contains all expressive means needed within this paper, but we will not consider $\mathcal{SHOQ}$ as such. Additional features of the yet more expressive DLs $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$ can be expressed by using $\mathcal{SHOQ}$ concepts in rules.

Definition 2. *Consider some DL* $\mathcal{L}$ *with concept expressions* $\mathbf{C}$, *individual names* N_I, *and role names* N_R, *and let* $\mathbf{V}$ *be a countable set of first-order variables. A* term *is an element of* $\mathbf{V} \cup \mathsf{N}_I$. *Given terms* t, u, *a* concept atom (role atom) *is a formula of the form* $C(t)$ $(R(t,u))$ *with* $C \in \mathbf{C}$ $(R \in \mathsf{N}_R)$.

A rule *for* $\mathcal{L}$ *is a formula* $B \rightarrow H$, *where* B *and* H *are conjunctions of (role and concept) atoms of* $\mathcal{L}$. *To simplify notation, we will often use finite sets* S *of atoms for representing the conjunction* $\bigwedge S$.

Semantically, rules are interpreted as first-order formulae, assuming that all variables are universally quantified, and using the standard first-order logic interpretation

Table 1. Semantics of concept constructors in $\mathcal{SHOQ}$ for an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$.

Name	Syntax	Semantics	Name	Syntax	Semantics
top	$\top$	$\Delta^{\mathcal{I}}$	nominal con.	$\{a\}$	$\{a^{\mathcal{I}}\}$
bottom	$\bot$	$\emptyset$	univ. rest.	$\forall U.C$	$\{x \in \Delta^{\mathcal{I}} \mid \langle x,y\rangle \in U^{\mathcal{I}} \text{ implies } y \in C^{\mathcal{I}}\}$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	exist. rest.	$\exists U.C$	$\{x \in \Delta^{\mathcal{I}} \mid y \in \Delta^{\mathcal{I}}\colon \langle x,y\rangle \in U^{\mathcal{I}}, y \in C^{\mathcal{I}}\}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	qualified	$\leq n\,R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \in \Delta^{\mathcal{I}} \mid \langle x,y\rangle \in R^{\mathcal{I}}, y \in C^{\mathcal{I}}\} \leq n\}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	number rest.	$\geq n\,R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \in \Delta^{\mathcal{I}} \mid \langle x,y\rangle \in R^{\mathcal{I}}, y \in C^{\mathcal{I}}\} \geq n\}$

of DL concepts (see Definition 3 below). In general, a DL knowledge base may entail the existence of *anonymous* domain elements that are not directly represented by some individual name, and it may even require models to be infinite. The fact that rules generally apply to all domain elements can therefore be problematic w.r.t. computability and complexity. It has thus been suggested to consider rules within which variables may only represent a finite amount of *named* individuals, i.e. individuals of the interpretation domain that are represented by some individual name in the knowledge base. Hence, effectively, these so-called *DL-safe* rules [7] apply to named individuals, but not to further anonymous individuals which have been inferred to exist.

Technically, this restriction can be achieved in various ways. The most common approach is to introduce a new concept expression HU that is asserted to contain the named individuals, and that is then used to restrict safe variables to that range. On the other hand, one can also dispense with this additional syntax by building the safety restriction directly into the semantics of variables – this is the intuition behind the use of *safe variables* in the following definition.

Definition 3. *An interpretation $\mathcal{I}$ consists of a set $\Delta^{\mathcal{I}}$ called* domain *(the elements of it being called* individuals*) together with a function $\cdot^{\mathcal{I}}$ mapping individual names to elements of $\Delta^{\mathcal{I}}$, concept names to subsets of $\Delta^{\mathcal{I}}$, and role names to subsets of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The function $\cdot^{\mathcal{I}}$ is inductively extended to role and concept expressions as shown in Table 1. An element $\delta \in \Delta^{\mathcal{I}}$ is a* named individual *if $\delta = a^{\mathcal{I}}$ for some $a \in \mathsf{N}_I$.*

Let $\mathbf{V}_S \subseteq \mathbf{V}$ be a fixed set of safe variables*. A* variable assignment *Z for an interpretation $\mathcal{I}$ is a mapping from the set of variables $\mathbf{V}$ to $\Delta^{\mathcal{I}}$ such that $Z(x)$ is named whenever $x \in \mathbf{V}_S$. Given a term $t \in \mathsf{N}_I \cup \mathbf{V}$, we set $t^{\mathcal{I},Z} := Z(t)$ if $t \in \mathbf{V}$, and $t^{\mathcal{I},Z} := t^{\mathcal{I}}$ otherwise. Given a concept atom $C(t)$ (role atom $R(t,u)$), we write $\mathcal{I}, Z \models C(t)$ ($\mathcal{I}, Z \models R(t,u)$) if $t^{\mathcal{I},Z} \in C^{\mathcal{I}}$ ($\langle t^{\mathcal{I},Z}, u^{\mathcal{I},Z}\rangle \in R^{\mathcal{I}}$), and we say that $\mathcal{I}$ and Z* satisfy *the atom in this case.*

An interpretation $\mathcal{I}$ satisfies *a rule $B \to H$ if, for all variable assignments Z for $\mathcal{I}$, either $\mathcal{I}$ and Z satisfy all atoms in H, or $\mathcal{I}$ and Z fail to satisfy some atom in B. In this case, we write $\mathcal{I} \models B \to H$ and say that $\mathcal{I}$ is a* model *for $B \to H$. An interpretation satisfies a set of rules (i.e. it is a* model *for this set) whenever it satisfies all elements of this set. A set of rules is* satisfiable *if it has a model, and* unsatisfiable *otherwise. Two sets of rules are* equivalent *if they have exactly the same models, and they are* equisatisfiable *if either both are unsatisfiable or both are satisfiable.*

Note that we have assumed earlier that N_I is always finite – typically it may comprise exactly the symbols that are actually used in the knowledge base –, and hence there are only a finite number of assignments for safe variables. Also note that empty

rule bodies are considered to be vacuously satisfied, and expressions of the form $\rightarrow H$ encode (sets of) facts. It is well-known that the satisfiability of sets of rules for DLs that support $\exists$ is undecidable, and we will introduce various restrictions to recover decidability below. One simple option is to restrict to so-called *Datalog* programs which we will later use to simulate inferences of more expressive rule languages:

Definition 4. *A rule is a* Datalog rule *if all concept atoms contained in it are of the form $C(t)$ with $C \in \mathsf{N}_C$, $\top(t)$, and $\bot(t)$. A* Datalog program *is a set of Datalog rules.*

3 DL Rules and ELP

In this section, we define the rule-based knowledge representation language ELP, and note that it subsumes several other existing languages in terms of expressivity. It is easy to see that unrestricted (SWRL) rules encompass even the very expressive DL $\mathcal{SROIQ}$ [1], since Tbox and Rbox axioms can readily be rewritten as rules. On the other hand, rules in their general form do not impose any of the restrictions on, e.g., *simple roles* or *regularity of Rboxes* that are crucial to retain decidability in $\mathcal{SROIQ}$. Recent works therefore have proposed *DL rules* as a decidable subset of SWRL that can be combined with various DLs without increasing the worst-case complexity of typical reasoning problems [9, 10].

We first recall DL rules (with conjunctions of simple roles) and apply them to the tractable DL $\mathcal{EL}^{++}$. The resulting formalism is the core of ELP, and significantly extends the expressivity of $\mathcal{EL}^{++}$ rules as considered in [9].

Definition 5. *Consider a rule $B \rightarrow H$ and terms $t, u \in \mathsf{N}_I \cup \mathbf{V}$. A* direct connection *from t to u is a non-empty set of atoms of the form $R(t, u)$. If B contains a direct connection between t and u, then t is* directly connected *to u. The term t is* connected *to u (in B) if the following inductive conditions apply:*

- *t is directly connected to u in B, or*
- *u is connected to t in B, or*
- *there is a variable $x \in \mathbf{V}$ such that t is connected to x and x is connected to u.*

An extended DL rule *is a rule $B \rightarrow H$ such that if variables $x \neq y$ in B are connected, then there is some direct connection $S \subseteq B$ such that x and y are not connected in $B \setminus S$.*

A path *from t to some variable x in B is a non-empty sequence of the form $R_1(x_1, x_2)$, $\ldots, R_n(x_n, x_{n+1}) \in B$ where $x_1 = t$, $x_2, \ldots, x_n \in \mathbf{V}$, $x_{n+1} = x$, and $x_i \neq x_{i+1}$ for $1 \leq i \leq n$. A term t in B is* initial *if there is no path to t. An extended DL rule is a* DL rule *if the following hold, where we assume x, y to range over variables $\mathbf{V}$, and t, t' to range over terms $\mathsf{N}_I \cup \mathbf{V}$:*

(1) for every variable x in B, there is a path from at most one initial term t to x,
(2) if $R(x, t) \in H$ or $C(x) \in H$, then x is initial in B,
(3) whenever $R(x, x) \in B$, we find that $R \in \mathsf{N}_R^{\mathsf{s}}$ is simple,
(4) whenever $R(t, x), R'(t, x) \in B$, we find that $R, R' \in \mathsf{N}_R^{\mathsf{s}}$ are simple,
(5) if $R(t, y) \in H$ with $R \in \mathsf{N}_R^{\mathsf{s}}$ simple, then all role atoms of the form $R'(t', y) \in B$ are such that $t' = t$ and $R' \in \mathsf{N}_R^{\mathsf{s}}$.

The above ensures that bodies of extended DL rules essentially correspond to sets of undirected trees, though reflexive "loops" $R(t, t)$ are also possible. Note that connections are essentially transitive but may not span over individual names. The notion of a connection turns out to be most convenient to establish the later decomposition of rules to accomplish the main tractability result in Theorem 14.

Bodies of DL rules are sets of directed trees due to item (1) in Definition 5. Two exceptions to that structure are admitted. Firstly, the definition of connections admits two elements of a path to be connected by multiple roles, corresponding to conjunctions of such roles. Secondly, atoms $R(x, x)$ are not taken into account for defining paths, such that local reflexivity conditions are admitted. Note that items (3) and (4) restricts both cases to simple roles.

Item (2) above ensures that the first variable in the rule head occurs in the rule body only as the root of some tree. Without this restriction, DL rules would be able to express inverse roles, even for DLs that deliberately exclude this feature to retain tractability. Extended DL rules waive requirements (1) and (2) to supply the expressivity of inverse roles, and indeed any extended DL rule that satisfies the additional requirements (3) to (5) on simplicity can be rewritten as a DL rule if inverse roles are available.

Item (5), finally, imposes the necessary restrictions on the use of simple roles, and, as an alternative presentation, one could also have *defined* the set of simple roles as the (unique) largest set of roles for which this requirement holds in a given rule base. In classical definitions of DLs, simple roles R are usually only admitted in role inclusion axioms of the form $S \sqsubseteq R$. Our definition relaxes this requirement to allow for further DL rules as long as these do not include certain role chains. For example, rules $C(x) \wedge D(y) \rightarrow R(x, y)$ and $R'(x, y) \wedge D(y) \rightarrow R(x, y)$ are possible even if R is simple.

We now apply DL rules to the description logic $\mathcal{EL}^{++}$ [2], for which many typical inference problems can be solved in polynomial time. We omit concrete domains in our presentation as they can basically be treated as shown in [2].

Definition 6. *An $\mathcal{EL}^{++}$ concept expression is a $\mathcal{SHOQ}$ concept expression that contains only the following concept constructors: $\sqcap$, $\exists$, $\top$, $\bot$, as well as nominal concepts $\{a\}$. An $\mathcal{EL}^{++}$* rule *is a DL rule for $\mathcal{EL}^{++}$, and an $\mathcal{EL}^{++}$* rule base *is a set of such rules.*

An $\mathcal{EL}^{++}$ knowledge base is a set of $\mathcal{EL}^{++}$ concept inclusions $C \sqsubseteq D$ and role inclusion axioms $R_1 \circ \ldots \circ R_n \sqsubseteq R$. See [2] for details. It is easy to see that any $\mathcal{EL}^{++}$ knowledge base can be written as an equivalent $\mathcal{EL}^{++}$ rule base. The above notion of $\mathcal{EL}^{++}$ rule bases extends [9] in two ways. Firstly, we now also allow conjunctions of simple roles, and secondly we allow atoms of the form $R(x, x)$ in rule bodies. Both extensions are non-trivial and require additional mechanisms during reasoning.

As we will see later, reasoning with $\mathcal{EL}^{++}$ rules is indeed possible in polynomial time. However, extending $\mathcal{EL}^{++}$ rules with further forms of rules, even if restricting to Datalog, readily leads to undecidability. This can be prevented if only *DL-safe* Datalog rules are permitted: a Datalog rule is DL-safe, if all of its variables are safe. Yet, this formalism can still capture all Datalog programs, and therefore satisfiability checking remains ExpTime hard [15].

Our strategy for extending $\mathcal{EL}^{++}$ rules into ELP therefore is to blend them with tractable fragments of DL-safe Datalog. As we will see below, one particular such Datalog fragment can again be characterised by the above notion of (extended) DL rule.

Another option is to allow only DL-safe Datalog rules of a particular form, namely those for which the number of variables per rule is bounded by some fixed finite number n. Indeed, it is easy to see that any DL-safe (Datalog) rule is equivalent to the set of rules obtained by replacing all safe variables by individual names in all possible ways. Since the replacements for each variable are independent, this leads to up to $|\mathsf{N}_I|^n$ different rules – which is a polynomial bound if n is a constant. Note, however, that large n might render practical computation infeasible.

In addition to various forms of DL-safe rules, ELP also allows for special rules of the form $R(x,y) \rightarrow C(y)$ expressing *range restrictions* on the role R. Such restrictions are neither DL-safe Datalog nor DL rules, and in general they do indeed lead to undecidability of $\mathcal{EL}^{++}$. However, it has recently been observed that range restrictions can still be admitted under certain conditions [6]. Therefore, even though this special form of rules is somewhat orthogonal to the other types of rules considered herein, we will include range restrictions into our considerations to give credit to their practical relevance.

Definition 7. *A rule $B \rightarrow H$ is a* basic ELP rule *if:*

- *$B \rightarrow H$ is an extended $\mathcal{EL}^{++}$ rule, and*
- *the rule $B' \rightarrow H'$ obtained from $B \rightarrow H$ by replacing all safe variables by some individual name is a DL rule.*

An ELP rule base RB *is a set of basic* ELP *rules together with* range restriction rules *of the form $R(x,y) \rightarrow C(y)$, that satisfies the following condition:*

- *If* RB *contains rules of the form $R(x,y) \rightarrow C(y)$ and $B \rightarrow H$ with $R(t,z) \in H$, then $C(z) \in B$.*

Whenever a set of range restriction rules satisfies the above condition for some set of ELP *rules, we say that the range restrictions are* admissible *for this rule set.*

A rule $B \rightarrow H$ is an ELP_n rule *for some natural number $n > 2$ if it is either an* ELP *rule, or a DL-safe Datalog rule with at most n variables.*

We remark that the above condition on admissibility of range restrictions is not quite the same as in [6]. Both versions ensure that, whenever an axiom entails some role atom $R(x,y)$, domain restrictions of R have no effect on the classification of y. The interaction of rules implying role atoms and range restrictions thus is strongly limited. In the presence of DL rules, we can accomplish this by restricting the applicability of rules by additional concept atoms $C(z)$ as in Definition 7. In [6], in contrast, additional range restrictions are required, and these, if added to an existing knowledge base, may also lead to new consequences. Any set of axioms that meets the requirements of [6] can clearly be extended to a semantically equivalent set of admissible ELP axioms, so that the approach of Definition 7 does indeed subsume the cases described in [6].

Before providing an extended example in the next section, we show how ELP subsumes some other tractable languages. One interesting case is DLP, a formalism introduced as the intersection of the DL $\mathcal{SHOIQ}$ and Datalog [4]. DLP can also be generalised using DL rules [9]: A *DLP head concept* is any $\mathcal{SHOQ}$ concept expression that includes only concept names, nominals, $\sqcap$, $\top$, $\bot$, and expressions of the form $\leq 1\, R.C$

where C is an $\mathcal{EL}^{++}$ concept expression. A *DLP rule* $B \rightarrow H$ is an extended DL rule such that all concept expressions in B are $\mathcal{EL}^{++}$ concept expressions, and all concept expressions in H are DLP head concepts.

Even the combination of DLP and $\mathcal{EL}$ contains the DL Horn-$\mathcal{FLE}$ and is thus ExpTime complete [5]. Yet, DLP and $\mathcal{EL}^{++}$ inferences can be recovered in ELP without losing tractability. In this sense, the following simple theorem substantiates our initial claim that ELP can be regarded as an extension both of DLP and $\mathcal{EL}^{++}$.

Theorem 8. *Consider any ground atom α of the form $C(a)$ or $R(a,b)$. Given a DLP rule base* RB *and an $\mathcal{EL}^{++}$ description logic knowledge base* KB*, one can compute an* ELP *rule base* RB′ *in linear time, such that: If* RB $\models \alpha$ *or* KB $\models \alpha$ *then also* RB′ $\models \alpha$*, and, if* RB′ $\models \alpha$ *then* RB $\cup$ KB $\models \alpha$*.*

Proof. The proof in [12] is based on observing that replacing all variables in a DLP rule base with safe variables does not affect satisfiability, since DLP does not infer the existence of new individuals. Now rules of the form $B \rightarrow \forall R.C(t)$ can be rewritten to $B \wedge R(t,y) \rightarrow C(y)$, and the result is easily seen to be in ELP given that all variables are safe. Rules of the form $B \rightarrow \leq 1\, R.C(t)$ are expressed by rules $B \wedge R(t,y_1) \wedge C(y_1) \wedge R(t,y_2) \wedge C(y_2) \rightarrow \approx_S (y_1,y_2)$, where $\approx_S$ is a new role for which the standard equality axioms (using safe variables) are added. The $\mathcal{EL}^{++}$ knowledge base can be added using the basic transformations given in the introduction (with new unsafe variables). □

Note that the resulting ELP rule base entails all individual consequences of RB and KB, and some but not all consequences of their (unsafe) union. ELP thus provides a means of combining $\mathcal{EL}^{++}$ and DLP in a way that prevents intractability, while still allowing for a controlled interaction between both languages. We argue that this is a meaningful way of combining both formalisms in practice since only some DLP axioms must be restricted to safe variables. Simple atomic concept and role inclusions, for example, can always be considered as $\mathcal{EL}^{++}$ axioms, and all concept subsumptions entailed from the $\mathcal{EL}^{++}$ part of a combined knowledge base do also affect classification of instances in the DLP part. DLP thus gains the terminological expressivity of $\mathcal{EL}^{++}$ while still having available specific constructs that may only affect the instance level.

4 Example

We now provide an extended example to illustrate the expressivity of ELP. The rules in Table 2 express a simplified conceptualisation of some preferences regarding food ordered in a restaurant: rule (1) states that all people that are allergic to nuts dislike all nut products, which is a kind of concept product. Rule (2) expresses the same for vegetarians and fish products. Rule (3) is a role conjunction, stating that anyone who ordered a dish he does not like will be unhappy. Rule (4) says that people generally dislike dishes that contain something that they dislike. Rule (5) is a range restriction for the role orderedDish. Rules (6) and (7) claim that any Thai curry contains peanut oil and some fish product, and the facts (8)–(12) assert various concept memberships.

We first verify that this is indeed a valid ELP rule base where all roles are simple. Indeed, the relaxed simplicity constraints on DL rules as given in Definition 5 are not

Table 2. A simple example rule base about food preferences. The variable v is assumed to be safe.

(1)	$\mathsf{NutAllergic}(x) \wedge \mathsf{NutProduct}(y) \rightarrow \mathsf{dislikes}(x, y)$
(2)	$\mathsf{Vegetarian}(x) \wedge \mathsf{FishProduct}(y) \rightarrow \mathsf{dislikes}(x, y)$
(3)	$\mathsf{orderedDish}(x, y) \wedge \mathsf{dislikes}(x, y) \rightarrow \mathsf{Unhappy}(x)$
(4)	$\mathsf{dislikes}(x, v) \wedge \mathsf{Dish}(y) \wedge \mathsf{contains}(y, v) \rightarrow \mathsf{dislikes}(x, y)$
(5)	$\mathsf{orderedDish}(x, y) \rightarrow \mathsf{Dish}(y)$
(6)	$\mathsf{ThaiCurry}(x) \rightarrow \mathsf{contains}(x, \mathsf{peanutOil})$
(7)	$\mathsf{ThaiCurry}(x) \rightarrow \exists \mathsf{contains.FishProduct}(x)$
(8)	$\rightarrow \mathsf{NutProduct}(\mathsf{peanutOil})$
(9)	$\rightarrow \mathsf{NutAllergic}(\mathsf{sebastian})$
(10)	$\rightarrow \exists \mathsf{orderedDish.ThaiCurry}(\mathsf{sebastian})$
(11)	$\rightarrow \mathsf{Vegetarian}(\mathsf{markus})$
(12)	$\rightarrow \exists \mathsf{orderedDish.ThaiCurry}(\mathsf{markus})$

violated in any of the rules. All rules other than (4) and (5) are readily recognised as $\mathcal{EL}^{++}$ rules. By first considering the connections in the respective rule bodies of (1)–(3), (6), and (7), we find that only rule (3) actually has connected terms at all, connected only by a single direct connection $\{\mathsf{orderedDish}(x, y), \mathsf{dislikes}(x, y)\}$. Both roles occurring in that connection are indeed simple. Similarly, the variable x is initial for these rules, and expressions of the form $R(z, z)$ do not occur.

It remains to check that also rules (4) and (5) are legal ELP statements. For rule (5), this requires us to check whether this range restriction rule is admissible, which is easy since no rule head contains atoms of the form $\mathsf{orderedDish}(t, y)$. For rule (4), we first need to check that it qualifies as an extended DL rule for $\mathcal{EL}^{++}$. This is easy to see since the direct connections in (4) do indeed form an undirected tree. Next, we assume that v was replaced by some individual name, and consider the paths in the rule. By Definition 5, paths must not end with individual names, and hence the modified rule contains no paths, such that it satisfies all conditions of an $\mathcal{EL}^{++}$ rule.

We can now investigate the semantics of the example. An interesting inference that can be made is $\mathsf{Unhappy}(\mathsf{sebastian})$. Indeed, combining (1), (8), and (9), we find that Sebastian dislikes peanut oil. Rule (10) implies that any interpretation must contain some domain element that is a Thai curry ordered by Sebastian, where we note that there is no individual name that explicitly refers to that curry. By (5) this unnamed curry is a dish, and by (6) it contains peanut oil. At this point we can apply rule (4), where v is mapped to the individual denoted by $\mathsf{peanutOil}$, x is mapped to the individual denoted by $\mathsf{sebastian}$, and y is mapped to the unnamed Thai curry. Hence we find that Sebastian dislikes his curry, and thus by rule (3) he is unhappy.

It is instructive to point out the use of safe and unsafe variables in that case. In contrast to plain Datalog, the above example involves computations relating to some unnamed individual – the Thai curry – to which rules are applied. On the other hand, rule (4) could only be invoked since the individual represented by v is named.

The impact of safety restrictions becomes clear by checking the happiness of Markus. Using similar inferences as above, we find that Markus ordered some (unnamed) Thai curry (12) – note that this need not be the same that was ordered by Sebastian – and

that this Thai curry contains some fish product (7) that Markus dislikes (2). However, this fish product is again unnamed, and hence we cannot apply rule (3), and we cannot conclude that Markus dislikes the dish he ordered. Thus, colloquially speaking, Markus is not unhappy since there is no information about some concrete (named) fish product in his curry.

5 Polytime ELP Reasoning with Datalog

We now introduce a polytime algorithm for compiling ELP rule bases into equisatisfiable Datalog programs. A useful feature of this transformation is that it does not only preserve satisfiability but also instance classification. Firstly, we observe that range restrictions in $\mathcal{EL}^{++}$ rule bases can be eliminated:

Proposition 9. *Consider an $\mathcal{EL}^{++}$ rule base* RB *and a set* RR *of range restrictions that are admissible for* RB. *Then there is a rule base* RB′ *that is equisatisfiable to* RB ∪ RR, *and which can be computed in polynomial time.*

The proof given in [12] extends the elimination strategy given in [6] to $\mathcal{EL}^{++}$ rules in a straightforward way. The main observation is that the formalisation of admissibility given above sufficiently generalises the conditions from [6] to encompass also concept-product-like rules that entail role relations without explicitly using roles in the antecedent. Next, we expand nested concept expressions in rules:

Definition 10. *An $\mathcal{EL}^{++}$ rule base* RB *is in* normal form *if all concept atoms in rule bodies are either concept names, $\top$, or nominals, all variables in a rule's head also occur in its body, and all rule heads contain only atoms of one of the following forms:*

$$A(t) \qquad \exists R.B(t) \qquad R(t,u)$$

where $A \in \mathsf{N}_C \cup \{\{a\} \mid a \in \mathsf{N}_I\} \cup \{\bot\}$, $B \in \mathsf{N}_C$, $R \in \mathsf{N}_R$, and $t, u \in \mathsf{N}_I \cup \mathbf{V}$.

Proposition 11. *Every $\mathcal{EL}^{++}$ rule base* RB *can be transformed in polynomial time into an equisatisfiable $\mathcal{EL}^{++}$ rule base* RB′ *in normal form.*

The following transformation of $\mathcal{EL}^{++}$ rules to Datalog is the core of our approach for reasoning in ELP:

Definition 12. *Given an $\mathcal{EL}^{++}$ rule base* RB *in normal form, the Datalog program* $\overline{\mathsf{P}}$(RB) *is defined as follows. The following new symbols are introduced:*

- *a role name $R_\approx$ (the* equality predicate*),*
- *concept names C_a for each $a \in \mathsf{N}_I$,*
- *concept names Self_R for each simple role $R \in \mathsf{N}_R^{\mathsf{s}}$,*
- *individual names $d_{R,C}$ for each $R \in \mathsf{N}_R$ and $C \in \mathsf{N}_C$.*

In the following, we will always use N_I, N_C, N_R, $\mathsf{N}_R^{\mathsf{n}}$, $\mathsf{N}_R^{\mathsf{s}}$ to refer to the original sets of symbols in RB, *not including the additional symbols added above. The program* $\overline{\mathsf{P}}$(RB) *is obtained from* RB *as follows:*

(a) For each individual name a occurring in RB*, the program* $\overline{\mathsf{P}}$(RB) *contains rules* $\rightarrow C_a(a)$ *and* $C_a(x) \rightarrow R_{\approx}(x, a)$.

(b) For each concept name C and role name R occurring in $\overline{\mathsf{P}}$(RB)*, the program* $\overline{\mathsf{P}}$(RB) *contains the rules*

$$\begin{array}{rr} \rightarrow R_{\approx}(x,x) & R(z,x) \wedge R_{\approx}(x,y) \rightarrow R(z,y) \\ R_{\approx}(x,y) \rightarrow R_{\approx}(y,x) & R(x,z) \wedge R_{\approx}(x,y) \rightarrow R(y,z) \\ C(x) \wedge R_{\approx}(x,y) \rightarrow C(y) & R_{\approx}(x,y) \wedge R_{\approx}(y,z) \rightarrow R_{\approx}(x,z) \end{array}$$

(c) For all rules $B \rightarrow H \in$ RB*, a rule $B' \rightarrow H' \in \overline{\mathsf{P}}$(RB) is created by replacing all occurrences of $R(x,x)$ by $\mathsf{Self}_R(x)$, all occurrences of $\{a\}(t)$ by $C_a(t)$, and all occurrences of $\exists R.C(t)$ with $C \in \mathsf{N}_C$ by the conjunction $R(t, d_{R,C}) \dot{\wedge} C(d_{R,C})$.*

(d) For all rules $B \rightarrow H \in$ RB *with $R(x,y) \in H$ and $R \in \mathsf{N}_R^{\mathsf{s}}$ simple,* $\overline{\mathsf{P}}$(RB) *contains a rule $B' \rightarrow \mathsf{Self}_R(x) \in \overline{\mathsf{P}}$(RB), where B' is obtained from B by replacing all occurrences of y with x, all occurrences of $\{a\}(t)$ by $C_a(t)$, and (finally) all expressions $S(x,x)$ with $\mathsf{Self}_S(x)$.*

(e) For each $R \in \mathsf{N}_R^{\mathsf{s}}$ and $a \in \mathsf{N}_I$, the rule $C_a(x) \wedge R(x,x) \rightarrow \mathsf{Self}_R(x)$ is in $\overline{\mathsf{P}}$(RB).

Theorem 13. *Given an $\mathcal{EL}^{++}$ rule base* RB *in normal form,* RB *is unsatisfiable iff* $\overline{\mathsf{P}}$(RB) *is unsatisfiable.*

Proof sketch. The proof in [12] proceeds by constructing models of $\overline{\mathsf{P}}$(RB) from models of RB, and vice versa. We omit the technical details here for space reasons, and merely sketch some of the relevant methods and insights.

It is well-known that, in the case of $\mathcal{EL}^{++}$, models can be generated by introducing only a single individual for each atomic concept [2]. For $\mathcal{EL}^{++}$ rules, however, the added features of role conjunction and local reflexivity change the situation: considering only one characteristic individual per atomic concept leads to undesired entailments in both cases. Our model constructions therefore deviate from the classical $\mathcal{EL}^{++}$ construction that worked for the simple $\mathcal{EL}$ rules in [9] with only minor modifications.

For instance, the rule base $\{a\}(x) \rightarrow \exists R.C(x), \{a\}(x) \rightarrow \exists S.C(x)$ does not entail any conjunction of the form $R(a,x) \wedge S(a,x)$. Yet, every interpretation in which the extension of C is a singleton set would necessarily entail this conjunction. This motivates the above use of $d_{R,C}$ in $\overline{\mathsf{P}}$(RB), which, intuitively, represent individuals of C that have been "generated" by a rule head of the form $\exists R.C(x)$. Thus we admit $|\mathsf{N}_R|$ distinct characteristic individuals for each class, and this suffices for the proper model construction in the presence of role conjunctions.

The second problematic feature are expressions of the form $R(x,x)$, which again preclude the consideration of only one characteristic individual per class. The use of concept atoms $\mathsf{Self}_R(x)$ enables the translation of models for RB to models of $\overline{\mathsf{P}}$(RB) (the soundness of the satisfiability checking algorithm). The latter may indeed entail additional statements of type $R(x,x)$ without impairing the validity of the Datalog rules that use $\mathsf{Self}_R(x)$.

In the other direction, models of RB are built from models of $\overline{\mathsf{P}}$(RB) by creating infinitely many "parallel copies" of a basic model structure. These copies form an infinite sequence of levels in the model, and simple roles relate only to successors in higher levels. Exceptions to this construction principle, such as the concept product rules discussed earlier, make the exact formalisation technically involved. The proof in [12] for

this case hinges upon the simplicity of roles in concepts Self_S, and it is not clear if a relaxation of this requirement would be possible. □

We are now ready to show the tractability of ELP.

Theorem 14. *Satisfiability of any* ELP_n *rule base* RB *can be decided in time polynomial in the size of* RB *and exponential in n. More precisely,* RB *can be transformed into an equisatisfiable Datalog program* P(RB) *which contains at most* $\max(3, n)$ *variables per rule, and this transformation is possible in polynomial time in the size of* RB. *Moreover, for any* $C \in \mathsf{N}_C$, $R \in \mathsf{N}_R$, *and* $a, b \in \mathsf{N}_I$, *we find that*

- $\mathrm{RB} \models C(a)$ *iff* $\mathsf{P}(\mathrm{RB}) \models C(a)$
- $\mathrm{RB} \models \{a\}(b)$ *iff* $\mathsf{P}(\mathrm{RB}) \models C_a(b)$
- $\mathrm{RB} \models R(a, b)$ *iff* $\mathsf{P}(\mathrm{RB}) \models R(a, b)$

Proof. We present some core parts of the proof in [12]. Grounding all safe variables of a rule base in all possible ways is a feasible reasoning method, but may lead to exponential increases in the size of the rule base. This can be prevented, however, by ensuring that any rule contains only a limited number of variables. A similar method can be used to ensure that the Datalog program $\overline{\mathsf{P}}(\cdot)$ as obtained in Definition 12 can be evaluated in polynomial time. We thus provide a satisfiability preserving polytime reduction of ELP rule bases into ELP rule bases that contain only a bounded number of variables per rule. We consider only basic ELP rules for the reduction, since range restrictions do not require any transformation. One should, however, observe that the transformation does not violate the admissibility restrictions for range restrictions.

Let $\mathrm{RB}' \subseteq \mathrm{RB}$ denote the set of ELP rules in RB (i.e. excluding only additional DL-safe rules of n variables that might be available in ELP_n). We first transform the ELP rule base into a normal form by applying the algorithm from Proposition 11. It is easy to see that this transformation can also be applied to ELP rules by treating safe variables like individual names. Hence, this transformation preserves satisfiability, and yields a rule base RB_1 the size of which is polynomial in the size of RB'. The new rule base RB_1 is then of a normal form similar to the one of Definition 10 but with additional safe components per rule.

Next, we reduce conjunctions in rule heads in the standard way: any rule of the form $B \to H_1 \wedge H_2$ is replaced by two rules $B \to H_1$ and $B \to H_2$ until all conjunctions in rule heads are eliminated. Again, the resulting rule base RB_2 is clearly equisatisfiable to RB_1 and can be obtained in polynomial time.

As the next step, we transform the extended DL rules of RB_2 into extended DL rules with at most 3 variables per rule. Besides the notions defined in Definition 5, we use a number of auxiliary notions in describing the transformation. In the following, we assume that all direct connections (cf. Definition 5) between terms t and u in some set B are *maximal*, i.e. contain all role atoms of the form $R(t, u) \in B$. Consider some rule $B \to H$:

- A *connected component* of B is a non-empty subset $S \subseteq B$ such that, for all terms $t \neq u$ occurring in S, we find that t and u are connected in S. A *maximal connected component* (MCC) is a connected component that has no supersets that are connected components.

- A variable x is *initial for* H if H is of the form $C(x)$ or $R(x, t)$.
- A variable x is *final for* H if H is of the form $R(t, x)$. If H is not of this form but $B \to H$ contains some variable, then some arbitrary but fixed variable in $B \to H$ is selected to be final for H.
- Given a subset S of B, we say that S is *reducible* if it contains variables that are neither initial nor final in H.
- Let S be an MCC of B, and consider a direct connection T from a term t to a term u in S. Let $S_{T,t}$ be the set of all atoms in S that contain some term t' connected to t in $S \setminus T$. Similarly, let $S_{T,u}$ be the set of all atoms in S that contain some term u' connected to u in $S \setminus T$.

Intuitively, the sets $S_{T,t}$ and $S_{T,u}$ consist of all atoms to the "left" or to the "right" of the connection T that can be reached from t and u, respectively, without using the atoms of T.

We can now proceed to reduce the forest structure of rule bodies.

In each iteration step of the reduction, select some rule $B \to H$ in RB_2 that contains more than three variables and some reducible MCC S of B, and do one of the following:

(1) If S contains no variable that is final for H, then select an initial element t as follows: if S contains a variable x that is initial for H then $t = x$; otherwise set $t = a$ for an arbitrary individual name $a \in \mathsf{N}_I$. The rule $B \to H$ is replaced by two new rules $(B \setminus S) \cup \{C(t)\} \to H$ and $S \to C(t)$, where C is a new concept name.

For all other cases, assume that the variable y in S is final for H.

(2) There is a direct connection T from y to some variable u such that $S_{T,u}$ is reducible but contains no variable initial for H. Then rule $B \to H$ is replaced by three new rules $B \cup \{C(y)\} \setminus (S_{T,u} \cup T) \to H$, $T \cup \{D(u)\} \to C(y)$, and $S_{T,u} \to D(u)$, where C, D are new concept names.

(3) There is a direct connection T from some variable t to y such that $S_{T,t}$ is reducible, and contains a variable x that is initial for H. Then rule $B \to H$ is replaced by three new rules $B \cup \{R(x, y)\} \setminus (S_{T,t} \cup T) \to H$, $\{R'(x, t)\} \cup T \to R(x, y)$, and $S_{T,t} \to R'(x, t)$, where R, R' are new non-simple role names.

(4) There is a direct connection T from some variable t to y such that $S_{T,t}$ is reducible but contains no variable that is initial for H. Then rule $B \to H$ is replaced by three new rules $B \cup \{R(a, y)\} \setminus (S_{T,t} \cup T) \to H$, $\{R'(a, t)\} \cup T \to R(a, y)$, and $S_{T,t} \to R'(a, t)$, where $a \in \mathsf{N}_I$ is an arbitrary individual name, and R, R' are new non-simple role names.

(5) There is a direct connection T from y to some variable u such that $S_{T,u}$ is reducible, and contains a variable x that is initial for H, and some further variable z besides x and u. We distinguish various cases:
   (a) There is a direct connection from some term $t \neq y$ to u. Then rule $B \to H$ is replaced by two new rules $B \cup \{R(x, u)\} \setminus S_{T,u} \to H$ and $S_{T,u} \to R(x, u)$, where R is a new non-simple role name.
   (b) The above is not the case, and there is some direct connection T' from u to some variable u' such that $S_{T',u'}$ is reducible but does not contain x. Then rule $B \to H$ is replaced by two new rules $B \cup \{C(u)\} \setminus (S_{T',u'} \cup T') \to H$ and $S_{T',u'} \cup T' \to C(u)$, where C is a new concept name.

(c) None of the above is the case, and u is involved in a direct connection T' besides T, which connects u to some variable u' such that $S_{T',u'}$ contains x. Let S_u denote the set $S_u := S \setminus (S_{T,y} \cup S_{T',u'})$. The rule $B \to H$ is replaced by two new rules $B \cup \{R(y,u')\} \setminus S_u \to H$ and $S_u \to R(y,u')$, with R a new non-simple role name.

This iteration is repeated until no further transformation is applicable. The proof in [12] proceeds by establishing various properties of the above reduction:

- All rules created in the above transformation are valid ELP rules.
- After the above translation, all rules in RB_2 have at most three variables in the body.
- The transformation terminates after a finite number of steps that is polynomially bounded in the size of RB_2.
- The above translation preserves satisfiability of RB_2.

Thus, the transformed rule base RB_2 is polynomial in the size of RB and contains at most three variables per rule. We can now compute the grounding of all safe variables in RB_2, i.e. the set of rules obtained by replacing safe variables in each rule of RB_2 with individual names in all possible ways. The obtained rule base is called RB_3 and its size clearly is polynomially bounded by $|\mathrm{RB}_2|^3$. Moreover, RB_3 is clearly equivalent to RB_2 and, by Definition 7, contains only $\mathcal{EL}^{++}$ rules and range restrictions. We can now apply the elimination of range restrictions of Proposition 9, and then use the normalisation from Proposition 11 to again obtain a set RB_4 of normalised $\mathcal{EL}^{++}$ rules. Again, RB_4 is equivalent to RB_3, and the transformations are easily seen to preserve the bound on the number of variables per rule, especially since rule bodies had already been normalised when computing RB_1.

Now, finally, the Datalog program $\bar{\mathsf{P}}(\mathrm{RB}_4)$ is constructed. By inspecting the cases of Definition 12, we find that $\bar{\mathsf{P}}(\mathrm{RB}_4)$ still contains at most 3 (unsafe) variables per rule. Since $\bar{\mathsf{P}}(\mathrm{RB}_4)$ and the initial set of basic ELP rules RB' are equisatisfiable, we can show that $\bar{\mathsf{P}}(\mathrm{RB}_4) \models C(a)$ iff $\mathrm{RB}' \models C(a)$ for all $C \in \mathsf{N}_C$ and $a \in \mathsf{N}_I$. The claim clearly holds if RB' is unsatisfiable. Otherwise, consider $\mathrm{RB}'' = \mathrm{RB}' \cup \{C(a) \to \bot(a)\}$, and again apply the above construction to obtain an according Datalog program $\bar{\mathsf{P}}(\mathrm{RB}''_4)$. Clearly, RB'' is unsatisfiable iff $\mathrm{RB}' \models C(a)$. But the former is equivalent to $\bar{\mathsf{P}}(\mathrm{RB}_4)$ being unsatisfiable. Since $\bar{\mathsf{P}}(\mathrm{RB}_4)$ is satisfiable, and since clearly $\bar{\mathsf{P}}(\mathrm{RB}''_4) = \bar{\mathsf{P}}(\mathrm{RB}_4) \cup \{C(a) \to \bot(a)\}$ (assuming that C and a occur in RB', and were thus already considered for the rules (a), (b), and (e) of $\bar{\mathsf{P}}(\mathrm{RB}_4)$), this is in turn equivalent to $\bar{\mathsf{P}}(\mathrm{RB}''_4) \models C(a)$ as claimed. In a similar fashion, one can show the correspondence for entailments of the form $\{a\}(b)$ ($C_a(b)$) and $R(a,b)$, similar to the statement claimed for the theorem.

The last result enables us to safely combine $\bar{\mathsf{P}}(\mathrm{RB}_4)$ with any additional DL-safe rule with n variables that may be present in ELP_n. For that purpose, one merely needs to introduce a concept HU and add facts $\to \mathsf{HU}(a)$ for all $a \in \mathsf{N}_I$. For each n-variable Datalog rule $B \to H$, a rule $B' \to H'$ then is created by replacing any atom of the form $\{a\}(t)$ by $C_a(t)$, and by adding a body atom $\mathsf{HU}(x)$ for any variable x occurring in $B \to H$. The resulting set of transformed Datalog rules is denoted LP, and we define $\mathsf{P}(\mathrm{RB}) := \bar{\mathsf{P}}(\mathrm{RB}_4) \cup LP$.

It is easy to see that $\mathsf{P}(\mathrm{RB})$ is equisatisfiable to RB, since RB' and $\bar{\mathsf{P}}(\mathrm{RB}_4)$ contain the corresponding ground facts, and since the rules of LP are applicable only to such

ground facts, where the above construction of *LP* establishes the required syntactic transformations and explicit safety conditions. Similarly, we also find that $\mathsf{P}(\mathrm{RB})$ entails the same ground facts as RB, as required in the theorem. Since $\overline{\mathsf{P}}(\mathrm{RB})$ is a Datalog program with at most $\max(3, n)$ variables per rule, it can naively be evaluated by computing its grounding, which is again bounded in size by $|\overline{\mathsf{P}}(\mathrm{RB})|^{\max(3,n)}$. Together with the polynomial size restrictions established for $\overline{\mathsf{P}}(\mathrm{RB})$, this shows the claimed worst-case complexity of reasoning. □

6 Discussion and Future Work

We have introduced ELP as a rule-based tractable knowledge representation language that generalises the known tractable description logics $\mathcal{EL}^{++}$ and DLP, where polynomial time reasoning was established using a novel reduction to Datalog. ELP in particular extends the DL $\mathcal{EL}^{++}$ with local reflexivity, concept products, conjunctions of simple roles, and limited range restrictions [6].

The notion of simple roles has been slightly extended as compared to the definition commonly used in DL, such that, e.g., the universal role can also be defined to be simple. A natural question is whether further extensions of ELP might be admissible. Regarding the simplicity restriction on role conjunctions, it is well-known that conjunctions of arbitrary roles in $\mathcal{EL}^{++}$ lead to undecidability. Querying for such conjunctions remains intractable [16] even when adopting regularity restrictions similar to the ones in $\mathcal{SROIQ}$. The complexity of using this feature in rules remains open, as does the question whether or not arbitrary roles could be used in reflexivity conditions of the form $R(x, x)$. The presented proofs, however, strongly depend on these restrictions.

The use of Datalog as an approach to solving DL reasoning tasks has been suggested in various works. KAON2 [8] provides an exponential reduction of $\mathcal{SHIQ}$ into disjunctive Datalog programs. The outcome of this reduction resembles our case since it admits for the easy extension with DL-safe rules and safe conjunctive queries. The model-theoretic relationships between knowledge base and Datalog program, however, are somewhat weaker than in our case. In particular, our approach admits queries for non-simple roles. Various other approaches used reductions to Datalog in order to establish mechanisms for conjunctive query answering [17–19]. These works differ from the presented approach in that they focus on general conjunctive query answering for $\mathcal{EL}$ and $\mathcal{EL}^{++}$, which is known to be more complex than satisfiability checking [16]. Another related approach is [20], where resolution-based reasoning methods for $\mathcal{EL}$ have been investigated (where we note that resolution is also the standard approach for evaluating Datalog). The methodology used there, however, is technically rather different from our presented approach.

Acknowledgements The authors wish to thank Carsten Lutz, Boris Motik, and Uli Sattler for useful discussions, and the anonymous reviewers for helpful comments. Research reported herein is supported by the EU in projects ACTIVE (IST-2007-215040) and NeOn (IST-2006-027595), and by the German Research Foundation under the ReaSem project.

References

1. 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 (KR2006), AAAI Press (June 2006) 57–67
2. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI-05), Edinburgh, UK, Morgan-Kaufmann Publishers (2005)
3. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. of Automated Reasoning **9** (2007) 385–429
4. Grosof, B., Horrocks, I., Volz, R., Decker, S.: Description logic programs: Combining logic programs with description logics. In: Proc. of WWW 2003, Budapest, Hungary, May 2003, ACM (2003) 48–57
5. Krötzsch, M., Rudolph, S., Hitzler, P.: Complexity of Horn description logics. In: Proc. 22nd AAAI Conf. (AAAI-07). (2007)
6. Baader, F., Lutz, C., Brandt, S.: Pushing the EL envelope further. In: Proc. 4th Int. Workshop on OWL: Experiences and Directions (OWLED-08 DC), Washington DC. (2008)
7. Motik, B., Sattler, U., Studer, R.: Query answering for OWL DL with rules. J. of Web Semantics **3** (2005) 41–60
8. Hustadt, U., Motik, B., Sattler, U.: Data complexity of reasoning in very expressive description logics. In: Proc. 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-05), Edinburgh, UK, Morgan-Kaufmann Publishers (2005) 466–471
9. Krötzsch, M., Rudolph, S., Hitzler, P.: Description logic rules. In: Proc. 18th European Conf. on Artificial Intelligence (ECAI-08), IOS Press (2008) 80–84
10. Gasse, F., Sattler, U., Haarslev, V.: Rewriting rules into $\mathcal{SROIQ}$ axioms. In: Poster at 21st Int. Workshop on Description Logics (DL-08). (2008)
11. Rudolph, S., Krötzsch, M., Hitzler, P.: All elephants are bigger than all mice. In: Proc. 21st Int. Workshop on Description Logics (DL-08). (2008)
12. Krötzsch, M., Rudolph, S., Hitzler, P.: ELP: Tractable rules for OWL 2. Technical report, Universität Karlsruhe, Germany (May 2008) `http://korrekt.org/page/ELP`.
13. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2007)
14. Horrocks, I., Patel-Schneider, P.F.: A proposal for an OWL rules language. In Feldman, S.I., Uretsky, M., Najork, M., Wills, C.E., eds.: Proc. 13th Int. Conf. on World Wide Web (WWW-04), ACM (2004) 723–731
15. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and expressive power of logic programming. ACM Computing Surveys **33** (2001) 374–425
16. Krötzsch, M., Rudolph, S., Hitzler, P.: Conjunctive queries for a tractable fragment of OWL 1.1. In Aberer, K., Choi, K.S., Noy, N., Allemang, D., Lee, K.I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P., eds.: Proc. 6th Int. Semantic Web Conference (ISWC 2007). Volume 4825 of LNCS., Springer (2007) 310–323
17. Pérez-Urbina, H., Motik, B., Horrocks, I.: Rewriting conjunctive queries under description logic constraints. In: Proc. Int. Workshop on Logic in Databases (LID-08). (2008)
18. Pérez-Urbina, H., Motik, B., Horrocks, I.: Rewriting conjunctive queries over description logic knowledge bases. In: Proc. Int. Workshop on Semantics in Data and Knowledge Bases (SDKB-08). (2008)
19. Rosati, R.: Conjunctive query answering in EL. In: Proc. 20th Int. Workshop on Description Logics (DL-07). (2007)
20. Kazakov, Y.: Saturation-Based Decision Procedures for Extensions of the Guarded Fragment. PhD thesis, Universität des Saarlandes, Germany (2005)