ELP: Tractable Rules for OWL 2[*]

Markus Krötzsch and **Sebastian Rudolph** and **Pascal Hitzler**
Institute AIFB, Universität Karlsruhe, Germany
{mak,sru,phi}@aifb.uni-karlsruhe.de

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.

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 $\mathcal{SROIQ}$ (Horrocks, Kutz, and Sattler 2006), 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 $\mathcal{EL}^{++}$ (Baader, Brandt, and Lutz 2005), DL Lite (Calvanese et al. 2007), and DLP (Grosof et al. 2003). 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 (Krötzsch, Rudolph, and Hitzler 2007a).

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 (Baader, Lutz, and Brandt 2008). 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 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 (Motik, Sattler, and Studer 2005). In these respects, the presented approach bears similarities with the KAON2 transformation of $\mathcal{SHIQ}$ knowledge bases into disjunctive Datalog programs (Hustadt, Motik, and Sattler 2005), though the actual algorithms of course 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 (Krötzsch, Rudolph, and Hitzler 2008; Gasse, Sattler, and Haarslev 2008). As shown in (Krötzsch, Rudolph, and Hitzler 2008), it is possible to indirectly express such rules by means of the expressive features provided by $\mathcal{SROIQ}$, 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 expressive features of ELP:

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

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

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 $\exists\,\mathsf{loves}.\mathsf{Self} \sqsubseteq \mathsf{Narcist}$

[*]Extended technical report with complete proofs

[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.

corresponds to $\mathsf{loves}(x, x) \rightarrow \mathsf{Narcist}(x)$.

Role disjointness Roles in $\mathcal{SROIQ}$ can be declared disjoint to state that elements 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 (Rudolph, Krötzsch, and Hitzler 2008). 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 classes, which is not directly possible in $\mathcal{SROIQ}$. An example is the rule $\mathsf{Woman}(x) \wedge \mathsf{hasChild}(x, y) \rightarrow \mathsf{motherOf}(x, y)$.

While this work is conceptually based on (Krötzsch, Rudolph, and Hitzler 2008), 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.

DLs, Rules, and DL-Safety

In this section, we introduce some basic notions of description logics (DL) (Baader et al. 2007). We will also consider rules that are logically similar to the Semantic Web Rule Language SWRL (Horrocks and Patel-Schneider 2004). Such rules may include DL concept expressions, and thus generalise the common DL axiom types of Abox, Tbox, and Rbox. We can therefore 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 symbols C, D will generally be used to denote 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 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 RB. Hence, effectively, these so-called *DL-safe* rules (Motik, Sattler, and Studer 2005) 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

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

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

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 element* if $\delta = a^{\mathcal{I}}$ for some $a \in \mathsf{N}_I$.

Let $\mathbf{V}_\mathsf{s} \subseteq \mathbf{V}$ 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 $x \in \mathbf{V}_\mathsf{s}$. 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 we find that $t^{\mathcal{I},Z} \in C^{\mathcal{I}}$ ($\langle t^{\mathcal{I},Z}, u^{\mathcal{I},Z}\rangle \in R^{\mathcal{I}}$). We say that $\mathcal{I}$ and Z *satisfy* the atom in this case, and we will omit Z whenever no variables occur.

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 RB –, 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 $\to H$ 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:

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.

We will later reduce sets of rules to Datalog so as to simulate inferences of more expressive rule languages within this simple formalism.

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 $\mathcal{SROIQ}$ (Horrocks, Kutz, and Sattler 2006), 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 (Krötzsch, Rudolph, and Hitzler 2008; Gasse, Sattler, and Haarslev 2008).

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 (Krötzsch, Rudolph, and Hitzler 2008).

Definition 5 Consider a rule $B \to H$ 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 $B \to H$ 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 $R_1(x_1, x_2), \ldots, R_n(x_n, x_{n+1}) \in B$ 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 16.

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 $C(x) \wedge D(y) \to R(x, y)$ and $R'(x, y) \wedge D(y) \to R(x, y)$ are possible even if R is simple.

We now apply DL rules to the description logic $\mathcal{EL}^{++}$ (Baader, Brandt, and Lutz 2005), 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 (Baader, Brandt, and Lutz 2005).

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 (Baader, Brandt, and Lutz 2005) 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 (Krötzsch, Rudolph, and Hitzler 2008) 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 (Dantsin et al. 2001).

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 $R(x, y) \to C(y)$ 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 (Baader, Lutz, and Brandt 2008). 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 $B \to H$ is a *basic* ELP *rule* if:

- $B \to H$ is an extended $\mathcal{EL}^{++}$ rule, and
- the rule $B' \to H'$ obtained from $B \to H$ 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 $R(x, y) \to C(y)$, that satisfies the following condition:

- If RB contains rules of the form $R(x, y) \to C(y)$ and $B \to H$ 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 $B \to H$ 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 (Baader, Lutz, and Brandt 2008). 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 (Baader, Lutz, and Brandt 2008), 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 (Baader, Lutz, and Brandt 2008) 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 (Baader, Lutz, and Brandt 2008).

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 (Grosof et al. 2003). DLP can also be generalised using DL rules (Krötzsch, Rudolph, and Hitzler 2008): A *DLP head concept* is any $\mathcal{SHOQ}$ concept expression that includes only concept names, nominal concepts, $\sqcap$, $\top$, $\bot$, and expressions of the form $\leq 1\, R.C$ where C is an $\mathcal{EL}^{++}$ concept expression. A *DLP rule* $B \to H$ 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 (Krötzsch, Rudolph, and Hitzler 2007a). 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 $\mathrm{RB} \models \alpha$ or $\mathrm{KB} \models \alpha$ then also $\mathrm{RB}' \models \alpha$, and, if $\mathrm{RB}' \models \alpha$ then $\mathrm{RB} \cup \mathrm{KB} \models \alpha$.

Proof. We assume that KB consists of Tbox and Rbox axioms as defined in (Baader, Brandt, and Lutz 2005) (Abox axioms can be internalised using nominal concepts as usual). We assume without loss of generality that the heads of all rules in RB are of the form $C(t)$, $\forall R.C$, and $\leq 1\, R.A$, where $C \in \mathsf{N}_C \cup \{\bot\}$ and A is an $\mathcal{EL}^{++}$ concept. This can easily be achieved by a simple transformation, similar to the one used in Proposition 11 later on.

In the following, we use a new auxiliary role name $\approx_S$. A new rule base $\hat{\mathrm{RB}}$ is obtained by first transforming rules of RB as follows:

- For each rule $B \to \forall R.C(t) \in \mathrm{RB}$, a rule $B \wedge R(t,y) \to C(t)$ is added to $\hat{\mathrm{RB}}$ where y is a new variable.
- For each rule $B \to\ \leq 1\, R.A(t) \in \mathrm{RB}$, a rule $B \wedge R(t,y_1) \wedge A(y_1) \wedge R(t,y_2) \wedge A(y_2) \to\ \approx_S(y_1,y_2)$ is added to $\hat{\mathrm{RB}}$ where y_1, y_2 are new variables.
- Each role $B \to C(t)$ that is not of the above forms is added to $\hat{\mathrm{RB}}$ without modifications.

The following equality theory is added to $\hat{\mathrm{RB}}$ for any role name R and concept name C in RB:

$$\begin{array}{rr} \to\ \approx_S(x,x) & R(z,x) \wedge \approx_S(x,y) \to\ R(z,y) \\ \approx_S(x,y) \to\ \approx_S(y,x) & R(x,z) \wedge \approx_S(x,y) \to\ R(y,z) \\ C(x) \wedge \approx_S(x,y) \to C(y) & \approx_S(x,y) \wedge \approx_S(y,z) \to\ \approx_S(x,z) \end{array}$$

Finally, all variables in $\hat{\mathrm{RB}}$ are uniformly replaced by safe variables.

It is not hard to see that RB and $\hat{\mathrm{RB}}$ are equisatisfiable. In particular, any DLP rule base that has a model, also has a model the domain of which is the set of individual names N_I, and that maps any such name to itself. It is a standard result of logic programming that Horn logic programs are satisfiable iff they admit such a least *Herbrand model*. As the restriction to safe variables does not affect these models, it is obvious that $\hat{\mathrm{RB}}$ has the same Herbrand models as RB, which establishes the claim. Moreover, it is easy to see that, for any $C \in \mathsf{N}_C$ and $a \in \mathsf{N}_I$, $\hat{\mathrm{RB}} \models C(a)$ iff $\mathrm{RB} \models C(a)$. A similar statement holds for role atoms $R(a,b)$.

Similarly, a rule base $\hat{\mathrm{KB}}$ is obtained by transforming the axioms of KB:

- For each Tbox axiom $C \sqsubseteq D \in \mathrm{KB}$, a rule $C(x) \to D(x)$ is added to $\hat{\mathrm{KB}}$.
- For each Rbox axiom $R_1 \circ \ldots \circ R_n \sqsubseteq R \in \mathrm{KB}$, a rule $R_1(x_1,x_2) \wedge \ldots \wedge R_n(x_n,x_{n+1}) \to R(x_1,x_{n+1})$ is added to $\hat{\mathrm{KB}}$.
- For each rule $B \to \forall R.C(t) \in \mathrm{RB}$, a rule $B' \to C(t)$ is added to $\hat{\mathrm{KB}}$

It is clear that KB and $\hat{\mathrm{KB}}$ are semantically equivalent, and thus KB and $\hat{\mathrm{KB}}$ entail the same ground atoms.

Now RB′ is defined to be the rule base $\hat{\mathrm{RB}} \cup \hat{\mathrm{KB}}$. It is easy to see that RB′ does indeed satisfy all requirements of Definition 7. By the above observations, any ground atom entailed by either RB or KB is also entailed by $\hat{\mathrm{RB}}$ or $\hat{\mathrm{KB}}$, respectively, and hence by RB′. Conversely, we find that $\mathrm{RB} \cup \mathrm{KB}$ semantically entails RB′, and thus any consequence of RB′ is also a consequence of $\mathrm{RB} \cup \mathrm{KB}$. □

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.

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

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

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

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 violated in any of the rules. All rules other than (4) and (5) are readily recognised as $\mathcal{EL}^{++}$ rules. By first considering the paths in the respective rule bodies, 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 cannot 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 peanutOil, x is mapped to the

(1)	$\mathsf{NutAllergic} \sqsubseteq \exists R_{\mathsf{NutAllergic}}.\mathsf{Self}$
	$\mathsf{NutProduct} \sqsubseteq \exists R_{\mathsf{NutProduct}}.\mathsf{Self}$
	$R_{\mathsf{NutAllergic}} \circ U \circ R_{\mathsf{NutProduct}} \sqsubseteq \mathsf{dislikes}$
(2)	$\mathsf{Vegetarian} \sqsubseteq \exists R_{\mathsf{Vegetarian}}.\mathsf{Self}$
	$\mathsf{FishProduct} \sqsubseteq \exists R_{\mathsf{FishProduct}}.\mathsf{Self}$
	$R_{\mathsf{Vegetarian}} \circ U \circ R_{\mathsf{FishProduct}} \sqsubseteq \mathsf{dislikes}$
(3)	$\exists(\mathsf{orderedDish} \sqcap \mathsf{dislikes}).\top \sqsubseteq \mathsf{Unhappy}$
(4)	$\mathsf{Dish} \sqsubseteq \exists R_{\mathsf{Dish}}.\mathsf{Self}$
	For all $v \in \mathsf{N}_I$:
	$\exists\mathsf{dislikes}.\{v\} \sqsubseteq \exists R_{\mathsf{dislikes\_}v}.\mathsf{Self}$
	$\exists\mathsf{contains}.\{v\} \sqsubseteq \exists R_{\mathsf{contains\_}v}.\mathsf{Self}$
	$R_{\mathsf{dislikes\_}v} \circ U \circ R_{\mathsf{Dish}} \circ R_{\mathsf{contains\_}v} \sqsubseteq \mathsf{dislikes}$
(5)	$\top \sqsubseteq \forall\mathsf{orderedDish}.\mathsf{Dish}$
(6)	$\mathsf{ThaiCurry} \sqsubseteq \exists\mathsf{contains}.\{\mathsf{peanutOil}\}$
(7)	$\mathsf{ThaiCurry} \sqsubseteq \exists\mathsf{contains}.\mathsf{FishProduct}$
(8)	$\mathsf{peanutOil} : \mathsf{NutProduct}$
(9)	$\mathsf{sebastian} : \mathsf{NutAllergic}$
(10)	$\mathsf{sebastian} : \exists\mathsf{orderedDish}.\mathsf{ThaiCurry}$
(11)	$\mathsf{markus} : \mathsf{Vegetarian}$
(12)	$\mathsf{markus} : \exists\mathsf{orderedDish}.\mathsf{ThaiCurry}$

Table 3: A DL knowledge base that expresses the semantics of the ELP rule base in Table 2. The concept expressions used are those of $\mathcal{EL}^{++}$ extended with reflexivity ($\exists R.\mathsf{Self}$), role conjunctions, and range restrictions. U denotes the universal role, and all role names with subscripts are auxiliary roles introduced only for the translation. See (Horrocks, Kutz, and Sattler 2006) for details on syntax and semantics of DL expressions not explained in this paper.

individual denoted by 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.

Using the correspondence between DL rules and description logics (Krötzsch, Rudolph, and Hitzler 2008), we can also rewrite the rule base of Table 2 into an equisatisfiable DL knowledge base. The result of this transformation is shown in Table 3. It should be noted that the knowledge base in the given form includes a role conjunction (3) that involves a non-simple role (dislikes) due to the use of auxiliary roles that represent concepts in role inclusion axioms.

The knowledge base therefore belongs to a DL that extends $\mathcal{SROIQ}$ with arbitrary role conjunctions – known to be undecidable in general. Since, in addition, it is not trivial to recover the intended rules from a DL knowledge base, rule-based representations in this case would be more suitable from an implementation point of view.

Note that besides (simple or non-simple) role conjunctions, $\mathcal{SROIQ}$ supports all constructs needed to represent ELP rule bases. However, quite some additional vocabulary is needed for representing rules, and especially safe variables introduce a significant (though polynomial) expansion of the knowledge base. Using an additional concept HU for named individuals may provide a slightly simpler approach. Yet, the DL representation of such complex roles is arguably less readable than the original ELP rule. From a user perspective, this motivates the use of rules as an auxiliary modelling metaphor when dealing with DL knowledge bases. Another useful addition in this respect might be the concept product operator as suggested in (Rudolph, Krötzsch, and Hitzler 2008), which could, e.g., be used to rewrite the axioms in (4) to

$$\exists\mathsf{dislikes}.\{v\} \times (\mathsf{Dish} \sqcap \exists\mathsf{contains}.\{v\}) \sqsubseteq \mathsf{dislikes},$$

for all $v \in \mathsf{N}_I$, thus avoiding much of the additional vocabulary. Similar expressions are possible to simplify the axioms of (1) and (2).

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.

Proof. The proof extends the elimination strategy given in (Baader, Lutz, and Brandt 2008) to $\mathcal{EL}^{++}$ rules in a straightforward way. The main observation is that the formalisation of admissibility given above sufficiently generalises the conditions from (Baader, Lutz, and Brandt 2008) to encompass also concept-product-like rules that entail role relations without explicitly using roles in the antecedent.

To construct the rule base RB′, we define the concept expression Range_R to be the (DL) conjunction of all concepts C that occur in an axiom of the form $R(x,y) \to C(y) \in \mathrm{RR}$. Now RB′ contains the following rules:

- for all R occurring in RR, and for all $a \in \mathsf{N}_I$, a rule $R(x,a) \to \mathsf{Range}_R(a)$,
- for all rules $B \to H$, a rule $B \to H'$, where H' is obtained by (recursively) replacing all (sub)concepts of the form $\exists R.C$ occurring in H by concepts $\exists R.(C \sqcap \mathsf{Range}_R)$.

Any model of RB ∪ RR is easily seen to be a model of RB′. For the other direction, consider some model $\mathcal{I}$ of RB′. The interpretation $\mathcal{I}$ may fail to be a model of RB ∪ RR if some axioms of RR are not satisfied, i.e., if there is some $R(x,y) \to C(y) \in \mathrm{RR}$ and some tuple $\langle\delta,\delta'\rangle \in R^{\mathcal{I}}$ such that $\delta' \notin C^{\mathcal{I}}$. However, we can repair $\mathcal{I}$ by simply removing all such tuples from the extension of R: an interpretation $\mathcal{I}'$ is defined to be equal to $\mathcal{I}$ regarding domain, interpretation of individual names, and interpretation of concept names, but with role names interpreted as follows:

$$R^{\mathcal{I}'} \coloneqq \{\langle\delta,\delta'\rangle \in R^{\mathcal{I}} \mid \delta' \in C^{\mathcal{I}} \text{ for all } R(x,y) \to C(y) \in \mathrm{RR}\}.$$

We claim that $\mathcal{I}'$ is a model of RB ∪ RR. By construction, $\mathcal{I}'$ clearly satisfies all axioms of RR. Now consider some rule $B \to H \in \mathrm{RB}$. Clearly, whenever $\mathcal{I}', Z \models B$ then also $\mathcal{I}, Z \models B$ and hence $\mathcal{I}, Z \models H'$, the rule head obtained from H in RB′. We consider the atoms that may occur in H:

- $R(t,u)$. If u is a variable, admissibility of RR requires B to contain Range_R which shows the claim. If u is an individual name, the rule $R(x,a) \to \mathsf{Range}_R(a)$ ensures entailment by $\mathcal{I}'$.
- $C(t)$. The differences between $\mathcal{I}'$ and $\mathcal{I}$ can only affect the entailment of $C(t)$ if C contains some subconcept $\exists R.D$. Since H' contains $\exists R.(D \sqcap \mathsf{Range}_R)$ in that case, it is easy to see that we do again find $\mathcal{I}', Z \models C(t)$ as required.

This shows that $\mathcal{I}', Z \models H$ and thus finishes the proof. □

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 nominal concepts, 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.

Proof. The transformation algorithm iteratively transforms RB. In each iteration, a rule $B \to H$ that is not in normal form yet is selected from RB, and one of the following is done:

- if H contains an atom of the form $(C \sqcap D)(t)$, then it is replaced by the conjunction $C(t) \wedge D(t)$,
- if H contains an atom of the form $\exists R.C(t)$ where $C \notin \mathsf{N}_C$, then this atom is replaced in H by $\exists R.A(t)$ with $A \in \mathsf{N}_C$ new, and a new rule $A(x) \to C(x)$ is added,
- if H contains an atom of the form $\top(t)$, then this atom is deleted from H. If H is singleton and would thus be empty after the deletion, then the whole rule $B \to H$ is deleted from RB,
- if H contains a variable x that is not contained in B, then the atom $\top(x)$ is added to B,

- if B contains an atom of the form $\exists R.C(t)$, it is replaced by two new atoms $R(t, y)$ and $C(y)$ where $y \in \mathbf{V}$ is new,
- if B contains an atom of the form $(C \sqcap D)(t)$, it is replaced by two new atoms $C(t)$ and $D(t)$,
- if B contains an atom of the form $\bot(t)$, then the rule $B \to H$ is deleted from RB.

It is easy to see that the transformation yields an equisatisfiable $\mathcal{EL}^{++}$ rule base in normal form, the size of which is polynomial in the size of the original rule base. □

The transformation of $\mathcal{EL}^{++}$ rules to Datalog is as follows:

Definition 12 Given an $\mathcal{EL}^{++}$ rule base RB in normal form, the Datalog program $\overline{\mathsf{P}}(\mathrm{RB})$ 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, N_R^n, N_R^s to refer to the original sets of symbols in RB, not including the additional symbols added above. The program $\overline{\mathsf{P}}(\mathrm{RB})$ is obtained from RB as follows:

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

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

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

(c) For all rules $B \to H \in \mathrm{RB}$, a rule $B' \to H' \in \overline{\mathsf{P}}(\mathrm{RB})$ 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 $R(t, d_{R,C}) \wedge C(d_{R,C})$.

(d) For all rules $B \to H \in \mathrm{RB}$ and $R(x, y) \in H$ with $R \in \mathsf{N}_R^\mathsf{s}$ simple, $\overline{\mathsf{P}}(\mathrm{RB})$ contains a rule $B' \to \mathsf{Self}_R(x) \in \overline{\mathsf{P}}(\mathrm{RB})$, 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 program $\overline{\mathsf{P}}(\mathrm{RB})$ contains the rule $C_a(x) \wedge R(x, x) \to \mathsf{Self}_R(x)$.

It is easy to see that $\overline{\mathsf{P}}(\mathrm{RB})$ is indeed a Datalog program. One can also readily construct the Datalog translation for the rules of Table 2, though the treatment of safe variables as in rule (4) will only be discussed further below. Most rules are already in Datalog and can remain unchanged by (c), with the exception of the existential entailments in (7), (10), and (12). The rule (7) is rewritten into a rule $\mathsf{ThaiCurry}(x) \to \mathsf{contains}(x, d_{\mathsf{contains},\mathsf{FishProduct}}) \wedge \mathsf{FishProduct}(d_{\mathsf{contains},\mathsf{FishProduct}})$. For rule (10), we additionally need to incorporate the range restriction (5) by using the reduction provided in Proposition 9. Thus we obtain the rule $\to \mathsf{orderedDish}(\mathsf{sebastian}, d_{\mathsf{orderedDish},\mathsf{ThaiCurry}}) \wedge \mathsf{ThaiCurry}(d_{\mathsf{orderedDish},\mathsf{ThaiCurry}})$. Rule (12) is treated similarly, and further auxiliary rules $\mathsf{orderedDish}(x, a) \to \mathsf{Dish}(a)$ are added as in Proposition 9. In addition, (d) generates new rules for (1) and (2) (we omit (4) here). Rule (1), e.g., leads to a rule $\mathsf{NutAllergic}(x) \wedge \mathsf{NutProduct}(x) \to \mathsf{Self}_{\mathsf{dislikes}}(x)$, which is clearly not relevant to reasoning in this case. Finally, the rules of (a), (b), and (e) are added to the program.

The correctness proof for this construction constitutes an essential part of the technical contributions of this paper, and we first provide some intuition on how the proof proceeds. To show that RB and $\overline{\mathsf{P}}(\mathrm{RB})$ are equisatisfiable, we construct models of $\overline{\mathsf{P}}(\mathrm{RB})$ from models of RB, and vice versa. It is well-known that, in the case of $\mathcal{EL}^{++}$, models can be generated by introducing only a single element for each atomic concept (Baader, Brandt, and Lutz 2005). For $\mathcal{EL}^{++}$ rules, however, the added features of role conjunction and local reflexivity change the situation: considering only one characteristic element 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 (Krötzsch, Rudolph, and Hitzler 2008) with only minor modifications.

For instance, the rule base $\{a\}(x) \to \exists R.C(x), \{a\}(x) \to \exists S.C(x)$ 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 $\overline{\mathsf{P}}(\mathrm{RB})$, which, intuitively, represent elements 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 concept, 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 concept. The use of concept atoms $\mathsf{Self}_R(x)$ enables the translation of models for RB to models of $\overline{\mathsf{P}}(\mathrm{RB})$ (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 $\overline{\mathsf{P}}(\mathrm{RB})$ 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 below proof 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.

Lemma 13 For any $\mathcal{EL}^{++}$ rule base RB in normal form, RB is unsatisfiable if $\overline{\mathsf{P}}(\mathrm{RB})$ is unsatisfiable.

Proof. We show the contrapositive: If RB is satisfiable, then so is $\overline{\mathsf{P}}(\mathrm{RB})$. Thus assume that RB has some model $\mathcal{I}$. We define an interpretation $\mathcal{J}$ of $\overline{\mathsf{P}}(\mathrm{RB})$ with domain $\Delta^\mathcal{J} = \mathsf{N}_I \cup \{d_{R,C} \mid R \in \mathsf{N}_R, C \in \mathsf{N}_C, C^\mathcal{I} \not\subseteq \{a\}^\mathcal{I} \text{ for all } a \in \mathsf{N}_I\}$. For any individual name d in $\overline{\mathsf{P}}(\mathrm{RB})$, set $d^\mathcal{J}$ as follows:

- If $d \in \Delta^{\mathcal{J}}$, then $d^{\mathcal{J}} = d$.
- Otherwise, if $d = d_{R,C}$ and $C^{\mathcal{I}} = \emptyset$, then $d^{\mathcal{J}} = a$ for some arbitrary $a \in \mathsf{N}_I$.
- Otherwise, if $d = d_{R,C}$ and $C^{\mathcal{I}} = \{a\}^{\mathcal{I}}$ for some $a \in \mathsf{N}_I$, then $d^{\mathcal{J}} = a$ for some (arbitrary) such a.

Moreover, we assign a concept expression $\kappa(\delta)$ to any element $\delta \in \Delta^{\mathcal{J}}$ as follows:

- if $\delta = a \in \mathsf{N}_I$ then $\kappa(\delta) = \{a\}$,
- if $\delta = d_{R,C}$ then $\kappa(\delta) = C$.

Now $\mathcal{J}$ interprets roles and concepts as follows (where we assume that C and R are symbols occurring in RB):

(A) $\delta \in C^{\mathcal{J}}$ iff $\kappa(\delta)^{\mathcal{I}} \subseteq C^{\mathcal{I}}$.

(B) $\delta \in C_a^{\mathcal{J}}$ iff $\kappa(\delta)^{\mathcal{I}} \subseteq \{a\}^{\mathcal{I}}$.

(C) $\delta \in \mathsf{Self}_R^{\mathcal{J}}$ iff $\langle \delta', \delta' \rangle \in R^{\mathcal{I}}$ for all $\delta' \in \kappa(\delta)^{\mathcal{I}}$.

(D) $\langle \delta, a \rangle \in R^{\mathcal{J}}$ for $a \in \mathsf{N}_I$ iff $\kappa(\delta)^{\mathcal{I}} \subseteq \exists R.\{a\}^{\mathcal{I}}$.

(E) $\langle \delta, d_{S,C} \rangle \in R^{\mathcal{J}}$ for $R \in \mathsf{N}_R^{\mathsf{n}}$ iff $\kappa(\delta)^{\mathcal{I}} \subseteq \exists R.C^{\mathcal{I}}$

(F) $\langle \delta, d_{S,C} \rangle \in R^{\mathcal{J}}$ for $R \in \mathsf{N}_R^{\mathsf{s}}$ and $\kappa(\delta)^{\mathcal{I}} \subseteq \exists S.C^{\mathcal{I}}$ iff $\langle \delta', \epsilon' \rangle \in R^{\mathcal{I}}$ for all $\delta' \in \kappa(\delta)^{\mathcal{I}}$ and $\epsilon' \in C^{\mathcal{I}}$ with $\langle \delta', \epsilon' \rangle \in S^{\mathcal{I}}$.

(G) $\langle \delta, d_{S,C} \rangle \in R^{\mathcal{J}}$ for $R \in \mathsf{N}_R^{\mathsf{s}}$ and $\kappa(\delta)^{\mathcal{I}} \not\subseteq \exists S.C^{\mathcal{I}}$ iff $\langle \delta', \epsilon' \rangle \in R^{\mathcal{I}}$ for all $\delta' \in \kappa(\delta)^{\mathcal{I}}$ and $\epsilon' \in C^{\mathcal{I}}$.

(H) $R_{\approx}^{\mathcal{J}} = \{\langle \delta, \delta \rangle \mid \delta \in \Delta^{\mathcal{J}}\}$

We claim that $\mathcal{J}$ is a model for $\overline{\mathsf{P}}(\text{RB})$. For the rules of type (a), (b), and (e) in Definition 12 this is easy to see. Now consider some rule $B' \to H'$ generated from a rule $B \to H \in \text{RB}$ by item (c). Assume there is a variable assignment Z' for $\mathcal{J}$ such that $\mathcal{J}, Z' \models B'$. We show how to iteratively construct a variable assignment Z for $\mathcal{I}$ such that $\mathcal{I}, Z \models B$:

As long as Z has not been defined for all variables occurring in B, do the following:

- Select a variable x occurring in B such that there is no atom $R(y, x) \in B$ with $y \neq x$ a variable for which Z is not defined yet. Note that such an x always exists, since $B \to H$ is a DL rule, and thus has no proper cycles.
- Select a value $Z(x) \in \kappa(Z'(x))^{\mathcal{I}}$ as follows:

(1) If $Z'(x) = a \in \mathsf{N}_I$ then set $Z(x) = a^{\mathcal{I}}$.

(2) Otherwise, if there is some $R(t, x) \in B'$ with $R \in \mathsf{N}_R^{\mathsf{n}}$, then let $Z(x)$ be some element $\delta \in \kappa(Z'(x))^{\mathcal{I}}$ such that $\langle t^{\mathcal{I},Z}, \delta \rangle \in R^{\mathcal{I}}$.

(3) Otherwise, we find that $Z'(x) = d_{S,C}$, and all role atoms in B' that contain x in the second position refer to simple roles. If there is some $R(t, x) \in B'$ such that $\kappa(t^{\mathcal{J},Z'})^{\mathcal{I}} \subseteq \exists S.C^{\mathcal{I}}$, then let $Z(x)$ be some element $\delta \in \kappa(Z'(x))^{\mathcal{I}}$ such that $\langle t^{\mathcal{I},Z}, \delta \rangle \in S^{\mathcal{I}}$.

(4) Otherwise let $Z(x)$ be some element $\delta \in \kappa(Z'(x))^{\mathcal{I}}$.

Finally, for all variables x not occurring in B, let $Z(x)$ be arbitrary.

We need to verify that Z is indeed well-defined. For that we must show that the choice of $Z(x)$ in the second item is always possible. To this end, note that $\kappa(Z'(x))^{\mathcal{I}}$ is non-empty by definition of κ. We check all cases separately:

(1) The given choice clearly is possible, and $Z(x) \in \kappa(Z'(x))^{\mathcal{I}}$.

(2) Since $R(t, x) \in B'$ with R non-simple, this atom is the only role atom with x in its second component by definition of DL rules (hence the choice of $R(t, x)$ is canonical). From $\mathcal{J}, Z' \models B'$ and (E) in the definition of $\mathcal{J}$ we conclude that $\kappa(t^{\mathcal{J},Z'})^{\mathcal{I}} \subseteq \exists R.\kappa(Z'(x))^{\mathcal{I}}$. By definition of Z (for case $t \in \mathbf{V}$) and $\mathcal{J}$ (for case $t \in \mathsf{N}_I$), we find that $t^{\mathcal{I},Z} \in \kappa(t^{\mathcal{J},Z'})$, and thus there must be a possible choice for $Z(x)$.

(3) In this case, the choice of $Z(x)$ depends on the term t in the first position of the selected atom $R(t, x)$. However, by the definition of DL rules, all atoms of the form $R'(t', x)$ must have the same term in their first component, and thus the choice of t is again canonical. By assumption, we find $\kappa(t^{\mathcal{J},Z'})^{\mathcal{I}} \subseteq \exists S.C^{\mathcal{I}}$, and we can apply a similar argument as in case (2) to conclude that the required choice of $Z(x)$ is possible.

(4) Trivial.

We further claim that $\mathcal{I}, Z \models B$, which is shown by considering all atoms that may occur in B:

- $C(t)$ with $C \in \mathsf{N}_C$. By Definition 12, B' also contains $C(t)$ and hence $\mathcal{J}, Z' \models C(t)$. If $t \in \mathbf{V}$ then, by construction of Z, we find that $Z(t) \in \kappa(Z'(t))^{\mathcal{I}}$. Hence, by item (A) in the definition of $\mathcal{J}$, $Z(t) \in C^{\mathcal{I}}$. Otherwise, if $t \in \mathsf{N}_I$ then we find that $\kappa(t)^{\mathcal{I}} = \{t\}^{\mathcal{I}} = \{t^{\mathcal{I}}\} \subseteq C^{\mathcal{I}}$ as required, where the subset inclusion follows again from (A).
- $\{a\}(t)$. This case is similar to the above item, with the only difference that $\{a\}$ corresponds to C_a in B', and that item (B) is used.
- $R(t, u)$. First assume that $u \in \mathbf{V}$. If $t = u$, then $\mathsf{Self}_R(u) \in B'$ and we can use (C) to conclude $\mathcal{I}, Z \models R(t, u)$. Otherwise, if $u \in \mathbf{V}$ and $t \neq u$, we can distinguish the cases as in the definition of Z:

(1) $\mathcal{I}, Z \models R(t, a)$ is a direct consequence of (D).

(2) The choice in case (2) of the definition of Z directly implies $\mathcal{I}, Z \models R(t, u)$, where it is important to note that only one such (non-simple) role atom with second argument u can occur.

(3) Again we have argued above that all role atoms with u in their second position must then be simple and refer to the same t in their first position. $Z(u)$ was chosen such that $\langle t^{\mathcal{I},Z}, Z(u) \rangle \in S^{\mathcal{I}}$. Therefore, $\mathcal{J}, Z' \models R(t, u)$ and (F) imply that $\langle t^{\mathcal{I},Z}, Z(u) \rangle \in R^{\mathcal{I}}$ as required.

(4) Case (G) in the definition of $\mathcal{J}$ applies, and hence we again conclude that $\langle t^{\mathcal{I},Z}, Z(u) \rangle \in R^{\mathcal{I}}$.

Finally, if $u \in \mathsf{N}_I$, then we can also apply the same reasoning as in case (1) above.

Thus find that $\mathcal{I}, Z \models B$, and, since $\mathcal{I}$ is assumed to be a model of RB, we conclude that $\mathcal{I}, Z \models H$. Moreover, for any variable x in B for which there is no atom $R(t, x) \in B$, and for any $\delta \in \kappa(Z'(x))^{\mathcal{I}}$, we can construct such a variable assignment Z which additionally satisfies $Z(x) = \delta$. This is easily seen from the definition of Z, the second item of which does not apply in that case.

We can now show that $\mathcal{J}, Z' \models H'$ by considering the different types of atoms that may occur in H. Using the notation of Definition 10, we have to consider three basic kinds of atoms: $A(t)$, $\exists R.B(t)$, and $R(t,u)$. If $t \in \mathbf{V}$ then, by the definition of DL rules, we find that there is no atom $R(u,t) \in B$ with $u \neq t$. Thus, for any $\delta \in \kappa(t^{\mathcal{J},Z'})^{\mathcal{I}}$, there is an assignment Z such that $t^{\mathcal{I},Z} = \delta$ and $\mathcal{I}, Z \models H$. This also is trivially true if $t \notin \mathbf{V}$, since $\kappa(t^{\mathcal{J},Z'})^{\mathcal{I}}$ contains only a single element $t^{\mathcal{I}}$ in this case. Using this insight (†), we can consider the various possible kinds of atoms in H:

- If $A(t) \in H$ with $A \in \mathsf{N}_C$ then also $A(t) \in H'$. Then (†) shows that $\delta \in A^{\mathcal{I}}$ for all $\delta \in \kappa(t^{\mathcal{J},Z'})^{\mathcal{I}}$, and we can conclude that $t^{\mathcal{J},Z'} \in A^{\mathcal{J}}$ by case (A) in the definition of $\mathcal{J}$.
- If $\{a\}(t) \in H$ then $C_a(t) \in H'$. We can apply the same reasoning as in the previous item, using (B) in the definition of $\mathcal{J}$.
- If $\exists R.B(t) \in H$ with $B \in \mathsf{N}_C \cup \{\top\}$ then $R(t, d_{R,B}) \wedge B(d_{R,B}) \in H'$. Firstly, this clearly entails that $B^{\mathcal{I}} \neq \emptyset$ and thus $d_{R,B} \in \Delta^{\mathcal{J}}$. By (A), we conclude that $d_{R,B} \in B^{\mathcal{J}}$, which establishes the second part of the above conjunction.

  To show that $R(t, d_{R,B})$ is also entailed, we again apply (†) as in the previous item to conclude that $\kappa(t^{\mathcal{J},Z'})^{\mathcal{I}} \subseteq \exists R.B^{\mathcal{I}}$. Thus we just need to observe that the conditions for $\mathcal{J}, Z' \models R(t, d_{R,B})$ that are given in (E) and (F), respectively, are satisfied.
- If $\exists R.B(t) \in H$ with $B = \{a\}$ then $R(t,a) \in H'$. Again, we can apply (†) to obtain $\kappa(t^{\mathcal{J},Z'})^{\mathcal{I}} \subseteq \exists R.\{a\}^{\mathcal{I}}$ which is all that is required to derive $\mathcal{J}, Z' \models R(t,a)$ from (D).
- If $R(t,u) \in H$ with $t \neq u$ then $R(t,u) \in H'$. Using (†) and the fact that $u^{\mathcal{I},Z} \in \kappa(u^{\mathcal{J},Z'})^{\mathcal{I}}$, we find that $\kappa(t^{\mathcal{J},Z'})^{\mathcal{I}} \subseteq \exists R.\kappa(u^{\mathcal{J},Z'})^{\mathcal{I}}$. This establishes the required conditions for (D) and (E), and thus settles all cases where either $u \in \mathsf{N}_I$, $u \in \mathbf{V}$ with $Z'(u) \in \mathsf{N}_I$, or $R \in \mathsf{N}_R^{\mathsf{n}}$.

  It remains to check the case where $u \in \mathbf{V}$ with $Z'(u) = d_{S,C}$ and $R \in \mathsf{N}_R^{\mathsf{s}}$. By the restrictions on simple roles in DL rules, we conclude that u occurs in the second position of role atoms in B' only if the atom is of the form $R'(t,u)$ with R' simple. If there is such an atom $R'(t,u) \in B'$ and if $\kappa(t^{\mathcal{J},Z'})^{\mathcal{I}} \subseteq \exists S.C^{\mathcal{I}}$, then the value for $Z(u)$ was chosen by case (3) of the definition of Z. We can thus derive a similar statement as (†), and conclude that $Z(u)$ might take any value $\epsilon' \in \kappa(Z'(u))^{\mathcal{I}}$ for which $\langle t^{\mathcal{I},Z}, \epsilon' \rangle \in S^{\mathcal{I}}$. Since we derive $\langle t^{\mathcal{I},Z}, \epsilon' \rangle \in R^{\mathcal{I}}$ in all these cases, we can invoke (F) to conclude $\mathcal{J}, Z' \models R(t,u)$.

  If there is no role atom $R'(t,u)$ in B', or if $\kappa(t^{\mathcal{J},Z'})^{\mathcal{I}} \not\subseteq \exists S.C^{\mathcal{I}}$ for all such atoms, then $Z(u)$ is chosen in case (4) of the definition of Z. A similar argument as before shows that the conditions of case (G) are satisfied in this case, and we obtain $\mathcal{J}, Z' \models R(t,u)$ as required.
- If $R(t,t) \in H$ then $\mathsf{Self}_R(t) \in H'$. Applying (†) again, we find that $\langle \delta, \delta \rangle \in R^{\mathcal{I}}$ for all $\delta \in \kappa(t^{\mathcal{J},Z'})^{\mathcal{I}}$. Using (C), we can again derive $\mathcal{J}, Z' \models \mathsf{Self}_R(t)$.

This shows that $\mathcal{J}, Z' \models H'$ and concludes the proof for rules of type (c).

Finally, for rules generated in item (d) of Definition 12, note that one could similarly obtain these rules by item (c) by adding, for each rule $B \to H \in \mathrm{RB}$ with $R(x,y) \in H$ and $R \in \mathsf{N}_R^{\mathsf{s}}$ simple, a rule $B' \to R(x,x)$, where B' is obtained from B by replacing all occurrences of y with x. Since adding such rules clearly does not affect the semantics of RB, case (d) is covered by case (c).

We conclude that $\mathcal{J}$ is indeed a model for all rules in $\mathsf{P}(\mathrm{RB})$ as required. □

Lemma 14 For any $\mathcal{EL}^{++}$ rule base RB in normal form, RB is unsatisfiable only if $\mathsf{P}(\mathrm{RB})$ is unsatisfiable.

Proof. The required constructions are very similar to the ones used in Lemma 13, yet some differences must be taken into account, and we therefore provide all arguments explicitly instead of referring to that earlier proof.

We show the contrapositive: If $\mathsf{P}(\mathrm{RB})$ is satisfiable then so is RB. Thus assume that $\mathcal{J}$ is a model of $\mathsf{P}(\mathrm{RB})$. We define an interpretation $\mathcal{I}$ of RB the domain $\Delta^{\mathcal{I}}$ of which is defined by setting $\Delta^{\mathcal{I}} = \mathsf{N}_I \cup \{d_{R,C,n} \mid R \in \mathsf{N}_R, C \in \mathsf{N}_C, n \geq 0, C^{\mathcal{J}} \not\subseteq \{a\}^{\mathcal{J}}$ for all $a \in \mathsf{N}_I\}$. Note that this ensures, in particular, that $C^{\mathcal{J}} \neq \emptyset$ for all elements $d_{R,C,n}$. The individual names in RB are interpreted in the obvious way by setting $a^{\mathcal{I}} = a$. We assign a concept expression $\kappa(\delta)$ to every element $\delta \in \Delta^{\mathcal{I}}$ as follows:

- if $\delta = a \in \mathsf{N}_I$ then $\kappa(\delta) = \{a\}$.
- if $\delta = d_{R,C,n}$ then $\kappa(\delta) = C$.

Note that by the conditions on elements of $\Delta^{\mathcal{I}}$, we find that $\kappa(\delta)^{\mathcal{J}} \neq \emptyset$ for all $\delta \in \Delta^{\mathcal{I}}$. We also assign a natural number $\nu(\delta)$ to each element: $\nu(d_{R,C,n}) = n$, and $\nu(a) = 0$ for all $a \in \mathsf{N}_I$. Now $\mathcal{I}$ interprets roles and concepts as follows:

(A) $\delta \in C^{\mathcal{I}}$ for $C \in \mathsf{N}_C$ iff $\kappa(\delta)^{\mathcal{J}} \subseteq C^{\mathcal{J}}$

(B) $\delta \in \{a\}^{\mathcal{I}}$ iff $\kappa(\delta)^{\mathcal{J}} \subseteq C_a^{\mathcal{J}}$

$\langle \delta, \epsilon \rangle \in R^{\mathcal{I}}$ iff one of the following holds:

(C) $\delta = d_{S,C,n}$, $\epsilon = d_{S,C,m}$ and $\kappa(\delta) \subseteq \mathsf{Self}_R^{\mathcal{J}}$.

(D) $\epsilon = a \in \mathsf{N}_I$ and $\kappa(\delta)^{\mathcal{J}} \subseteq \exists R.\{a\}^{\mathcal{J}}$

(E) $\epsilon = d_{S,C,n}$, $R \in \mathsf{N}_R^{\mathsf{n}}$, and $\kappa(\delta)^{\mathcal{J}} \subseteq \exists R.C^{\mathcal{J}}$

(F) $\epsilon = d_{S,C,n}$, $R \in \mathsf{N}_R^{\mathsf{s}}$, $\kappa(\delta)^{\mathcal{J}} \subseteq \exists S.C^{\mathcal{J}}$, $\nu(\delta) < n$, and $\langle \delta', \epsilon' \rangle \in R^{\mathcal{J}}$ for all $\delta' \in \kappa(\delta)^{\mathcal{J}}$ and $\epsilon' \in C^{\mathcal{J}}$ with $\langle \delta', \epsilon' \rangle \in S^{\mathcal{J}}$.

(G) $\langle \delta', \epsilon' \rangle \in R^{\mathcal{J}}$ for all $\delta' \in \kappa(\delta)^{\mathcal{J}}$ and $\epsilon' \in \kappa(\epsilon)^{\mathcal{J}}$.

Note how we use the additional numerical index n in (F) to ensure that anonymous elements do never relate to themselves unless explicitly needed. In contrast to Lemma 13, case (B) now implies that $\delta = a^{\mathcal{I}}$. Thus, cases (A) and (B) have some dependency: whenever $C^{\mathcal{I}} \neq \emptyset$ and $C^{\mathcal{I}} \subseteq \{a\}^{\mathcal{I}}$ we also must have that $a^{\mathcal{I}} \in C^{\mathcal{I}}$, or otherwise $\mathcal{I}$ is not well-defined. Fortunately, this condition is indeed satisfied: since $\mathcal{J}$ satisfies the rules $C_a(x) \to xR_{\approx}a$ of (a), we find that $\langle \delta, a^{\mathcal{J}} \rangle \in R_{\approx}{}^{\mathcal{J}}$ for all $\delta \in C^{\mathcal{J}}$. But then, using the rules of (b), we can conclude $a^{\mathcal{J}} \in C^{\mathcal{I}}$ as required.

We claim that $\mathcal{I}$ is a model of RB. Thus consider any rule $B \to H$ in RB and let $B' \to H' \in \mathsf{P}(\mathrm{RB})$ be the rule obtained from $B \to H$ by item (c) of Definition 12. Assume that there is a variable assignment Z such that $\mathcal{I}, Z \models B$. We show how to iteratively construct a variable assignment Z' such that $\mathcal{J}, Z' \models B'$:

As long as Z' has not been defined for all variables occurring in B', do the following:

- Select a variable x occurring in B' such that there is no atom $R(y, x) \in B'$ where y is a variable for which Z' is not defined yet. Note that such an x always exists, since $B' \to H'$ is a DL rule, and thus has no proper cycles.
- Select a value $Z'(x) \in \kappa(Z(x))^{\mathcal{J}}$ as follows:

(1) If $Z(x) = a \in \mathsf{N}_I$ then set $Z'(x) = a^{\mathcal{I}}$.

(2) Otherwise, if there is some $R(t, x) \in B'$ with $R \in \mathsf{N}_R^{\mathsf{n}}$, then let $Z'(x)$ be some element $\delta \in \kappa(Z(x))^{\mathcal{J}}$ such that $\langle t^{\mathcal{J},Z'}, \delta\rangle \in R^{\mathcal{J}}$.

(3) Otherwise, we find that $Z(x) = d_{S,C,n}$, and all role atoms in B' that contain x in the second position refer to simple roles. If there is some $R(t, x) \in B'$ such that $\nu(t^{\mathcal{I},Z}) < \nu(Z(x))$ and $\kappa(t^{\mathcal{I},Z})^{\mathcal{J}} \subseteq \exists S.C^{\mathcal{J}}$, then let $Z'(x)$ be some element $\delta \in \kappa(Z(x))^{\mathcal{J}}$ such that $\langle t^{\mathcal{J},Z'}, \delta\rangle \in S^{\mathcal{J}}$.

(4) Otherwise, if there is some $R(t, x) \in B'$ such that $t^{\mathcal{I},Z}$ if of the form $d_{S,C,m}$ and $\kappa(t^{\mathcal{I},Z})^{\mathcal{J}} \subseteq \mathsf{Self}_R^{\mathcal{J}}$, then let $Z'(x)$ be some element $t^{\mathcal{J},Z'}$.

(5) Otherwise let $Z'(x)$ be some element $\delta \in \kappa(Z(x))^{\mathcal{J}}$.

Finally, for all variables x not occurring in B, let $Z'(x)$ be arbitrary.

We need to verify that Z' is indeed well-defined. For that we must show that the choice of $Z'(x)$ in the second item is always possible. To this end, note that $\kappa(Z(x))^{\mathcal{J}}$ is non-empty by definition of κ and $\Delta^{\mathcal{I}}$. We check all cases separately:

(1) The given choice clearly is possible, and $Z'(x) \in \kappa(Z(x))^{\mathcal{J}}$.

(2) Since $R(t, x) \in B'$ with R non-simple, this atom is the only role atom with x in its second component by definition of DL rules (hence the choice of $R(t, x)$ is canonical). From $\mathcal{I}, Z \models B'$ and (E) in the definition of $\mathcal{I}$ we conclude that $\kappa(t^{\mathcal{I},Z})^{\mathcal{J}} \subseteq \exists R.\kappa(Z(x))^{\mathcal{J}}$. By definition of Z' (for case $t \in \mathbf{V}$) and $\mathcal{I}$ (for case $t \in \mathsf{N}_I$), we find that $t^{\mathcal{J},Z'} \in \kappa(t^{\mathcal{I},Z})$, and thus there must be a possible choice for $Z'(x)$.

(3) In this case, the choice of $Z'(x)$ depends on the term t in the first position of the selected atom $R(t, x)$. However, by the definition of DL rules, all atoms of the form $R'(t', x)$ must have the same term in their first component, and thus the choice of t is again canonical. By assumption, we find $\kappa(t^{\mathcal{I},Z})^{\mathcal{J}} \subseteq \exists S.C^{\mathcal{J}}$, and we can apply a similar argument as in case (2) to conclude that the required choice of $Z'(x)$ is possible.

(4) Trivial.

(5) Trivial.

We further claim that $\mathcal{J}, Z' \models B'$, which is shown by considering all atoms that may occur in B':

- $C(t)$ with $C \in \mathsf{N}_C$ occurring in RB. By Definition 12, B also contains $C(t)$ and hence $\mathcal{I}, Z \models C(t)$. If $t \in \mathbf{V}$ then, by construction of Z', we find that $Z'(t) \in \kappa(Z(t))^{\mathcal{J}}$. Hence, by item (A) in the definition of $\mathcal{I}$, $Z'(t) \in C^{\mathcal{J}}$. Otherwise, if $t \in \mathsf{N}_I$ then we find that $\kappa(t)^{\mathcal{J}} = \{t\}^{\mathcal{J}} = \{t^{\mathcal{J}}\} \subseteq C^{\mathcal{J}}$ as required, where the subset inclusion follows again from (A).
- $C_a(t)$. This case is similar to the above item, with the only difference that C_a corresponds to $\{a\}$ in B, and that item (B) is used.
- $\mathsf{Self}_R(t)$. By construction of B', we find that $t \in \mathbf{V}$ and $R(t, t) \in B$. If $Z'(t) \notin \mathsf{N}_I$, we can use (C) to conclude $\mathcal{J}, Z' \models \mathsf{Self}_R(t)$. Otherwise, case (D) applies and we can conclude that $\langle t^{\mathcal{J}}, t^{\mathcal{J}}\rangle \in R^{\mathcal{J}}$. Now since $\mathcal{J}$ satisfies the rules generated in item (e), we can conclude that $\mathcal{J}, Z' \models \mathsf{Self}_R(t)$ as required.
- $R(t, u)$ with R occurring in RB. If $u \in \mathbf{V}$ we find that $t \neq u$, and we can distinguish the cases as in the definition of Z':

(1) $\mathcal{J}, Z' \models R(t, a)$ is a direct consequence of (D).

(2) The choice in case (2) of the definition of Z' directly implies $\mathcal{J}, Z' \models R(t, u)$, where it is important to note that only one such (non-simple) role atom with second argument u can occur.

(3) Again we have argued above that all role atoms with u in their second position must then be simple and refer to the same t in their first position. $Z'(u)$ was chosen such that $\langle t^{\mathcal{J},Z'}, Z'(u)\rangle \in S^{\mathcal{J}}$. Therefore, $\mathcal{I}, Z \models R(t, u)$ and (F) imply that $\langle t^{\mathcal{J},Z'}, Z'(u)\rangle \in R^{\mathcal{J}}$ as required.

(4) Using a similar argumentation as in the previous item, we conclude that $\langle t^{\mathcal{J},Z'}, Z'(u)\rangle \in R^{\mathcal{J}}$ by item (C).

(5) Given that the conditions of (1)–(4) did not hold, $\mathcal{I}, Z \models R(t, u)$ can only follow from case (G) in the definition of $\mathcal{I}$ applies. Hence we again conclude that $\langle t^{\mathcal{J},Z'}, Z'(u)\rangle \in R^{\mathcal{J}}$.

Finally, if $u \in \mathsf{N}_I$, then we can also apply the same reasoning as in case (1) above.

Thus find that $\mathcal{J}, Z' \models B'$, and, since $\mathcal{J}$ is assumed to be a model of RB, we conclude that $\mathcal{J}, Z' \models H'$. Moreover, for any variable x in B' for which there is no atom $R(t, x) \in B'$, and for any $\delta \in \kappa(Z(x))^{\mathcal{J}}$, we can construct such a variable assignment Z' which additionally satisfies $Z'(x) = \delta$. This is easily seen from the definition of Z', the second item of which does not apply in that case.

Note that the rules obtained in item (d) can all be generated by (d) when adding a suitable rule to RB, where this rule clearly does not change the semantics of RB, and hence our argument extends to all rules of (d) as well.

We can now show that $\mathcal{I}, Z \models H$ by considering the different types of atoms that may occur in H'. Atoms in H' can have two basic forms: $C(t)$ and $R(t, u)$. If $t \in \mathbf{V}$ then, by the definition of DL rules, we find that there is no atom $R'(u', t) \in B$ with $u' \neq t$. Thus, for any $\delta \in \kappa(t^{\mathcal{I},Z})^{\mathcal{J}}$, there is an assignment Z' such that $t^{\mathcal{J},Z'} = \delta$ and $\mathcal{J}, Z' \models H'$. This also is trivially true if $t \notin \mathbf{V}$, since $\kappa(t^{\mathcal{I},Z})^{\mathcal{J}}$ contains only a single element $t^{\mathcal{J}}$ in this case. Using this insight (†), we can consider the various possible kinds of atoms in H (and thus H'):

- If $C(t) \in H$ with $C \in \mathsf{N}_C$ then also $C(t) \in H'$. Then (†) shows that $\delta \in C^{\mathcal{J}}$ for all $\delta \in \kappa(t^{\mathcal{I},Z})^{\mathcal{J}}$, and we can conclude that $t^{\mathcal{I},Z} \in C^{\mathcal{I}}$ by case (A) in the definition of $\mathcal{I}$.

- If $\{a\}(t) \in H$ then $C_a(t) \in H'$. We can apply the same reasoning as in the previous item, using (B) in the definition of $\mathcal{J}$.
- If $R(t,u) \in H$ with $t \neq u$ then $R(t,u) \in H'$. Using (†) and the fact that $u^{\mathcal{J},Z'} \in \kappa(u^{\mathcal{I},Z})^{\mathcal{J}}$, we find that $\kappa(t^{\mathcal{I},Z})^{\mathcal{J}} \subseteq \exists R.\kappa(u^{\mathcal{I},Z})^{\mathcal{J}}$. This establishes the required conditions for (D) and (E), and thus settles all cases where either $u \in \mathsf{N}_I$, $u \in \mathbf{V}$ with $Z(u) \in \mathsf{N}_I$, of $R \in \mathsf{N}_R^{\mathsf{n}}$.

  It remains to check the case where $u \in \mathbf{V}$ with $Z(u) = d_{S,C,n}$ and $R \in \mathsf{N}_R^{\mathsf{s}}$. By the restrictions on simple roles in DL rules, we conclude that u occurs in the second position of role atoms in B' only if the atom is of the form $R'(t,u)$ with R' simple. We first assume that there is one or more such atom $R'(t,u) \in B'$ and distinguish various cases:
  - $\nu(t^{\mathcal{I},Z}) \geq n$. Since we have $\mathcal{I}, Z \models R'(t,u)$, one of the cases (C) – (G) must apply to $\langle t^{\mathcal{I},Z}, Z(u)\rangle$ and R', for all such atoms $R'(t,u)$. Cases (D), (E), and (F) are precluded by our assumptions.

    Now if $t^{\mathcal{I},Z}$ is not of the form $d_{S,C,m}$, then case (C) is also precluded. From case (G) we conclude that $\langle \delta', \epsilon'\rangle \in R'^{\mathcal{J}}$ for all $\delta' \in \kappa(t^{\mathcal{I},Z})^{\mathcal{J}}$ and $\epsilon' \in \kappa(Z(u))^{\mathcal{J}}$. Using an argument similar to (†), we find that a variable assignment Z' with $\mathcal{J}, Z' \models B' \to H'$ can be constructed such that $Z'(u) = \epsilon'$ for any $\epsilon' \in \kappa(Z(u))^{\mathcal{J}}$, since case (5) of the construction of Z' applies. Thus $\langle \delta, \epsilon\rangle \in R^{\mathcal{J}}$ for all $\delta \in \kappa(t^{\mathcal{I},Z})^{\mathcal{J}}$ and $\epsilon \in \kappa(Z(u))^{\mathcal{J}}$, and we can apply (G) to conclude $\mathcal{I}, Z \models R(t,u)$.

    Alternatively, $t^{\mathcal{I},Z}$ is of the form $d_{S,C,m}$. If $\kappa(t^{\mathcal{I},Z})^{\mathcal{J}} \nsubseteq \mathsf{Self}_R^{\mathcal{J}}$ then case (G) applies as above. Otherwise, $\mathcal{I}, Z \models R'(t,u)$ might be concluded by case (C), and $Z'(u)$ was defined by case (4). Now by the rules (d) of $\mathsf{P}(\mathrm{RB})$, we find that there must exist a rule based on $B \to H$ that has the form $B'' \to \mathsf{Self}_R(t)$, where we note that the assumptions on $t^{\mathcal{I},Z}$ imply that t is a variable. By the construction in (d), it is easy to see that $\mathcal{J}, Z' \models B'' \to \mathsf{Self}_R(t)$ and hence we can conclude that $t^{\mathcal{J},Z'} \in \mathsf{Self}_R^{\mathcal{J}}$. Applying (†), we finally find that the conditions of (C) are indeed satisfied, and that $\mathcal{I}, Z \models R(t,u)$ can be concluded.
  - $\nu(t^{\mathcal{I},Z}) < n$. If $\kappa(t^{\mathcal{I},Z})^{\mathcal{J}} \subseteq \exists S.C^{\mathcal{J}}$, then the value for $Z'(u)$ was chosen by case (3) of the definition of Z'. We can thus again derive a similar statement as (†), and conclude that $Z'(u)$ might take any value $\epsilon' \in \kappa(Z(u))^{\mathcal{J}}$ for which $\langle t^{\mathcal{J},Z'}, \epsilon'\rangle \in S^{\mathcal{J}}$. Since we derive $\langle t^{\mathcal{J},Z'}, \epsilon'\rangle \in R^{\mathcal{J}}$ in all these cases, we can invoke (F) to conclude $\mathcal{I}, Z \models R(t,u)$.

    If $\kappa(t^{\mathcal{I},Z})^{\mathcal{J}} \nsubseteq \exists S.C^{\mathcal{J}}$ for all such atoms, then $Z'(u)$ is chosen in case (4) or (5) of the definition of Z'. Case (4) can be treated as above, and a similar argument also shows that the conditions of case (G) are satisfied in case (5). Thus we obtain $\mathcal{I}, Z \models R(t,u)$ as required.

  Finally, if there is no role atom $R'(t,u)$ in B', then again case (G) can be invoked to derive the desired result.
- If $\exists R.B(t) \in H$ then H' contains an according conjunction of role and concept atoms, and we can apply the same arguments as above to see that similar statements hold for $\mathcal{I}$, from which the required result can be concluded.
- If $R(t,t) \in H$ then $\mathsf{Self}_R(t) \in H'$. Applying (†) again, we find that $\kappa(t^{\mathcal{I},Z})^{\mathcal{J}} \subseteq \mathsf{Self}_R^{\mathcal{J}}$ and we can apply (C) to derive $\mathcal{I}, Z \models R(t,t)$.

This shows that $\mathcal{I}, Z \models H$ in all cases, and thus concludes the proof. □

Summing up the result of Lemma 13 and Lemma 14, we obtain the following next theorem:

Theorem 15 Given an $\mathcal{EL}^{++}$ rule base RB in normal form, RB is unsatisfiable iff $\mathsf{P}(\mathrm{RB})$ is unsatisfiable.

Next, we want to show that ELP is indeed tractable. The above results on $\mathcal{EL}^{++}$ rules already provide a way of deciding satisfiability of ELP by first grounding safe variables, and then proceeding with the elimination of range restrictions and transformation to Datalog. When grounding safe variables into individual names, however, we must first ensure that this grounding does not incur an exponential blow-up of the rule base. Moreover, we also need to show that the resulting Datalog program can be evaluated in polynomial time.

The proof thus proceeds by decomposing ELP rules into rules containing a limited finite number of (safe or unsafe) variables. The grounding of safe variables then can only produce a polynomially bounded number of new rules. After translating from $\mathcal{EL}^{++}$ rules to Datalog, the number of variables per rule is still bounded. Since Datalog contains no existential quantifiers, Datalog programs are equivalent to their grounding, i.e. we can again replace variables with individual names in all possible ways. Note that this time, the relevant individual names for grounding also include new symbols of the form $d_{R,C}$. Evaluating the resulting variable-free Datalog program is P-complete.

The decomposition of ELP rules into rules with a bounded number of variables exploits the forest shape of rule bodies by iteratively reducing branches of trees. This is not possible for general n-variable Datalog rules in ELP_n, which thus increase the number of rules exponentially in n. Yet, admitting rules of some small n might well be feasible in practice, and we therefore include them into the following theorem.

Theorem 16 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 $\mathsf{P}(\mathrm{RB})$ 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. 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 knowledge 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 $\underline{\mathsf{P}}(\cdot)$ as obtained in Definition 12 can be evaluated in polynomial time. We therefore proceed by providing 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 lead to a violation on 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 $S_u \to R(y,u')$, where R is a new non-simple role name.

This iteration is repeated until no further transformation is applicable. It is easy to see that the translation preserves conditions on simplicity of roles, since all newly introduced roles are non-simple, and since they do never occur in a body position where simplicity is required.

In all considerations below, we will use the notation of the above cases when considering some transformation step, and refer to the generated rules in each step by the order of their appearance in the transformation steps (e.g. by saying "first rule of (2)" or "rule 3 of (4)").

Claim 1 All rules created in the above transformation are valid ELP rules.

Most cases directly follow from the fact that subsets of rule bodies of ELP rules satisfy most of the requirements of Definition 5 and Definition 7. An additional check is required to verify that, for some new rule head $C(x)$ or $R(x,t)$ with x unsafe, x is indeed initial in the body. This is readily verified for the rules in (1), and for the first two rules generated in (2), given that one minds the direction of the atoms in T. For the rule 3 of (2), problems might occur only if u is an unsafe variable. But in this case it is clearly initial in $S_{T,u}$: if it would have a direct connection from some element t other than y, then both t and y would either be initial or have a path from some initial element. But the initial elements for t and y cannot be the same without violating the condition for an extended DL rule, and hence u would have paths from two distinct initial element, which in turn contradicts the conditions on DL rules. This shows that rule 3 of (2) is also a valid ELP rule. Cases (3), (4), and all cases of (5) are again immediate.

Further care must be taken when introducing auxiliary roles, since the first condition of DL rules (paths from unique initial elements) might be violated whenever an auxiliary role creates additional paths to some variable. New role atoms are introduced in (3) and (4), but in each case only to either replace an existing direct connection to the variable y (first rules), or as part of a "chain" of role atoms (rules 2). Similar observations can be made in case (5)(c). For case (5)(a), note that the precondition implies that u already is the target of direct connections from two distinct terms y and t. Thus, u cannot be an unsafe variable, and the reduction is permissible, even though it clearly leads to multiple direct connections leading to u in rule 1.

Claim 2 After the above translation, all rules in RB_2 have at most three variables in the body.

First note that when looking for reducible sets of the form $S_{T,u}$ for some direct connection T and term u, it is in order to restrict to the case where u is a variable. This is so, since S is assumed to be an MCC, and hence any variable in $S_{T,u}$ is connected to the variables in $S \setminus S_{T,u}$ as well, but connections are not transitive over individual names. Hence the variable must also be part of some set $S_{T,z}$ with z being a variable.

For a contradiction, suppose that there is some rule $B \to H$ with at least four variables in B. By assumption, none of the cases of the translation is applicable to that rule. However, there must be some reducible MCC S in B. Otherwise, B would contain no variables besides the initial and final one, contradicting our assumption. Thus let S be some reducible component in B. Since rule (1) is not applicable, we can assume that S contains a final variable y.

Since S is reducible, some atom of S contains a variable that is neither final nor initial for H. Since cases (3) and (4) are not applicable, we conclude that there is no direct connection T from some variable t to y such that $S_{T,t}$ is reducible. But since S is a connected component, all terms of S are connected to y, and hence there must be a direct connection T from y to some variable u such that $S_{T,u}$ is reducible. Since (2) does not apply, T must be such that $S_{T,u}$ contains the initial variable x. Since only one such T can exist (due to the tree shape asserted for extended DL rules), and since $B \to H$ contains more than three variables by assumption, some additional variable z besides x and u must exist in $S_{T,u}$, and thus the preconditions of case (5) hold.

It remains to show that one of the three sub-cases of (5) must apply. Assuming that (a) and (b) do not hold, we conclude that there is no direct connection from any term $t' \neq y$ to u, and that there is no direct connection T' from u to some term u' such that $S_{T',u'}$ is reducible and does not contain x. Yet we know that u is directly connected with some term other than t, since $S_{T,u}$ contains a variable besides x (i.e. is reducible). Therefore there is some connection T' from u to some term u' such that $S_{T',u'}$ contains x, and (c) is indeed applicable.

Claim 3 The transformation terminates after a finite number of steps that is polynomially bounded in the size of RB_2.

For any set S of atoms, let $\nu(S)$ be the number of (safe or unsafe) variable names in S. Given a rule $B \to H \in \mathrm{RB}_2$, a number $\gamma(B \to H)$, called the *reduction number* of $B \to H$, is then defined by setting $\gamma(B \to H) := \max(0, \nu(B \cup H) - 3)$. Moreover, $\gamma(\mathrm{RB}_2)$ is defined as the sum of $\gamma(B \to H)$ for all $B \to H \in \mathrm{RB}_2$. Clearly, $\gamma(\mathrm{RB}_2)$ is polynomially bounded by the size of RB_2.

We claim that the above transformation terminates after at most $\gamma(\mathrm{RB}_2)$ steps. Clearly, no transformation can be applied if $\gamma(\mathrm{RB}_2) = 0$. It remains to show that, whenever RB_2' is obtained from RB_2 by any of the transformation steps, we find that $\gamma(\mathrm{RB}_2) > \gamma(\mathrm{RB}_2')$. This is achieved by considering all transformations individually. The technical difficulty in this part arises from the individual $\max(\cdot)$ computations involved in γ: even if a rule gets smaller, this might not equally reduce its reduction number, since there are no negative reduction numbers. In other words, each rule may contain up to three variables that do not count. We will sometimes assume that those three have been selected for some rule and speak of "non-counting variables" and "counting variables."

For case (1), note that S contains some variable that is neither final nor initial for H, and that $B \to H$ has at least 4 variables. We may thus assume that S contains a counting variable. Therefore rule 1 has at least one counting variable less than $B \to H$. If $\nu(S) \leq 3$, then rule 2 has reduction number 0 and the claim follows. If $\nu(S) > 3$ then we may assume that S contains at most two non-counting variables of B, since $B \to H$ also contains some variable y final for H that is not contained in S. Hence rule 1 has at least $\nu(S) - 2$ counting variables less. Rule 2 in turn has only $\nu(S) - 3$

counting variables, so that the claim follows again.

For case (2), we obtain three new rules. Rule 2 clearly has at most two distinct terms and hence no counting variables. We use n to denote $\nu(S_{T,u})$, the number of variables in $S_{T,u}$. Since $S_{T,u}$ is reducible, $n \geq 1$. Again, since there are 4 or more variables in $B \to H$, we can assume that $S_{T,u}$ contains at least one variable that is counting in $B \to H$. The reduction number of rule 1 therefore is strictly smaller than $\gamma(B \to H)$, and this suffices whenever $n \leq 3$ (since the reduction number of rule 3 is 0 in that case). Now assume that $n > 3$. Since y can be assumed to be non-counting, $S_{T,u}$ contains at most 2 non-counting variables of B, and hence rule 1 has at least $n - 2$ counting variables less. Rule 3, in turn, has only $n - 3$ non-counting variables, which again proves the overall reduction.

Cases (3) and (4) can be shown by a similar argumentation. Again, rule 2 does not add to the overall reduction number in either case, and the sum of rules 1 and 3 is found to decrease by a case distinction as above. Cases (5)(a) and (5)(b) are also similar, though there are only two rules in this case. Note that for (a) the additional variable z in $S_{T,u}$ is strictly required to obtain a reduction. For case (5)(c), the result follows since u is assumed to be a variable, so that again the reduction number of the transformed rule 1 decreases (while the other rule has at most three variables).

Claim 4 The above translation preserves satisfiability of RB_2.

This can be shown by a simple induction, given that all possible transformation steps preserve satisfiability. This is generally rather easy to see, but we show one case formally for illustration. Thus consider transformation step (1), where $B \to H$ is the considered rule, and $B_1 \to H$ and $B_2 \to C(t)$ denote the generated rules. Clearly, adding $B_2 \to C(t)$ to RB_2 preserves satisfiability since C is new. Thus it remains to show equisatisfiability of $\mathrm{RB}_2' \coloneqq \mathrm{RB}_2 \cup \{B_2 \to C(t)\}$ and $\mathrm{RB}_2'' \coloneqq \mathrm{RB}_2 \cup \{B_2 \to C(t), B_1 \to H\} \setminus \{B \to H\}$.

Thus consider some interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathrm{RB}_2'$. Then there is some interpretation $\mathcal{I}'$ with $\mathcal{I}' \models \mathrm{RB}_2'$ and $C^{\mathcal{I}'} = \{\delta \in \Delta^{\mathcal{I}'} \mid \mathcal{I}', Z \models B_2$ for some variable assignment Z with $t^{\mathcal{I}',Z} = \delta\}$. A suitable $\mathcal{I}'$ can be obtained from $\mathcal{I}$ by minimising the extent of C while preserving all other aspects of the interpretation, which can be done since C is new. Note that $\mathcal{I}' \models B_2 \to C(t)$ by definition. We claim that $\mathcal{I}' \models \mathrm{RB}_2''$. Thus assume that $\mathcal{I}', Z \models B_1$ for some variable assignment Z. Then $\mathcal{I}', Z \models C(t)$ and thus $t^{\mathcal{I}',Z} \in C^{\mathcal{I}'}$. By the assumptions on $C^{\mathcal{I}'}$, we find that there is some variable assignment Z' such that $\mathcal{I}', Z' \models B_2$ where $t^{\mathcal{I}',Z} = t^{\mathcal{I}',Z'}$. Now observe that, by construction, B_2 and B_1 contain no common variables, other than possibly t (if t is a variable). Thus there is some variable assignment Z'' such that $Z''(x) = Z(x)$ for any variable x in B_1 and $Z''(x) = Z'(x)$ for any variable x in B_2. But then $\mathcal{I}', Z'' \models B_1 \cup B_2$. As defined in (1), $(B_1 \cup B_2) \supseteq B$ and thus $\mathcal{I}', Z'' \models B$, and we can conclude $\mathcal{I}', Z'' \models H$ since $\mathcal{I}' \models B \to H$. By definition, Z and Z'' agree on all terms in H and thus we obtain $\mathcal{I}', Z \models H$ as required. Since Z was arbitrary, this shows that $\mathcal{I} \models B_1 \to H$, and hence $\mathcal{I}' \models \mathrm{RB}_2''$.

For the other direction, consider some interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathrm{RB}_2''$. We claim that $\mathcal{I} \models \mathrm{RB}_2'$. Thus assume that $\mathcal{I}, Z \models B$ for some variable assignment Z. Then also $\mathcal{I}, Z \models B_2$ as $B_2 \subseteq B$, and hence $\mathcal{I}, Z \models C(t)$. But then $\mathcal{I}, Z \models B_1$ and thus $\mathcal{I}, Z \models H$ as required.

The cases (2)–(5) can be treated in a similar fashion, where again it is essential that each case completely eliminates certain terms from the transformed rule, so that the required merging of variable assignments Z' and Z'' is indeed possible.

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 $\overline{\mathsf{P}}(\mathrm{RB}_4)$ is constructed. By inspecting the cases of Definition 12, we find that $\overline{\mathsf{P}}(\mathrm{RB}_4)$ still contains at most 3 (unsafe) variables per rule. Since $\overline{\mathsf{P}}(\mathrm{RB}_4)$ and the initial set of basic ELP rules RB' are equisatisfiable, we can show that $\overline{\mathsf{P}}(\mathrm{RB}_4) \models C(a)$ 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 $\overline{\mathsf{P}}(\mathrm{RB}_4'')$. Clearly, RB'' is unsatisfiable iff $\mathrm{RB}' \models C(a)$. But the former is equivalent to $\overline{\mathsf{P}}(\mathrm{RB}_4)$ being unsatisfiable. Since $\overline{\mathsf{P}}(\mathrm{RB}_4)$ is satisfiable, and since clearly $\overline{\mathsf{P}}(\mathrm{RB}_4'') = \overline{\mathsf{P}}(\mathrm{RB}_4) \cup \{C(a) \to \bot(a)\}$ (assuming that C and a occur in RB', and were thus already considered for the rules (a), (b), and (e) of $\overline{\mathsf{P}}(\mathrm{RB}_4)$), this is in turn equivalent to $\overline{\mathsf{P}}(\mathrm{RB}_4'') \models C(a)$ 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 $\overline{\mathsf{P}}(\mathrm{RB}_4)$ 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 $\mathsf{P}(\mathrm{RB}) \coloneqq \overline{\mathsf{P}}(\mathrm{RB}_4) \cup LP$.

It is easy to see that $\mathsf{P}(\mathrm{RB})$ is equisatisfiable to RB, since RB' and $\overline{\mathsf{P}}(\mathrm{RB}_4)$ 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 $\underline{\mathsf{P}}(\mathrm{RB})$ 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 $|\underline{\mathsf{P}}(\mathrm{RB})|^{\max(3,n)}$. Together with the polynomial size restrictions established for $\underline{\mathsf{P}}(\mathrm{RB})$, this shows the claimed worst-case complexity of reasoning. □

We remark that one could also have deferred the grounding of safe variables in ELP rules in the above proof by using the auxiliary predicate HU for such variables as well, instead of replacing them by individual names before further translation. This would be appropriate in practice, but it would complicate the above proof since this form of replacement would not lead to an $\mathcal{EL}^{++}$ rule base to which Definition 12 could readily be applied.

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 (Baader, Lutz, and Brandt 2008).

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 (Krötzsch, Rudolph, and Hitzler 2007b) 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 (Hustadt, Motik, and Sattler 2005) 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 (Pérez-Urbina, Motik, and Horrocks 2008b; 2008a; Rosati 2007). 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 (Krötzsch, Rudolph, and Hitzler 2007b). Another related approach is (Kazakov 2005), 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 Boris Motik, Carsten Lutz, and Uli Sattler for useful hints and discussions during the production of this paper, 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

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P., eds. 2007. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the EL envelope. In *Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI-05)*. Edinburgh, UK: Morgan-Kaufmann Publishers.

Baader, F.; Lutz, C.; and Brandt, S. 2008. Pushing the EL envelope further. In *Proc. 4th Int. Workshop on OWL: Experiences and Directions (OWLED-08 DC), Washington DC*.

Calvanese, D.; Giacomo, G. D.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. of Automated Reasoning* 9:385–429.

Dantsin, E.; Eiter, T.; Gottlob, G.; and Voronkov, A. 2001. Complexity and expressive power of logic programming. *ACM Computing Surveys* 33:374–425.

Gasse, F.; Sattler, U.; and Haarslev, V. 2008. Rewriting rules into $\mathcal{SROIQ}$ axioms. In *Poster at 21st Int. Workshop on Description Logics (DL-08)*.

Grosof, B.; Horrocks, I.; Volz, R.; and Decker, S. 2003. Description logic programs: Combining logic programs with description logics. In *Proc. of WWW 2003, Budapest, Hungary, May 2003*, 48–57. ACM.

Horrocks, I., and Patel-Schneider, P. F. 2004. A proposal for an OWL rules language. In Feldman, S. I.; Uretsky, M.; Najork, M.; and Wills, C. E., eds., *Proc. 13th Int. Conf. on World Wide Web (WWW-04)*, 723–731. ACM.

Horrocks, I.; Kutz, O.; and Sattler, U. 2006. The even more irresistible $\mathcal{SROIQ}$. In *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2006)*, 57–67. AAAI Press.

Hustadt, U.; Motik, B.; and Sattler, U. 2005. Data complexity of reasoning in very expressive description logics. In *Proc. 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-05)*, 466–471. Edinburgh, UK: Morgan-Kaufmann Publishers.

Kazakov, Y. 2005. *Saturation-Based Decision Procedures for Extensions of the Guarded Fragment*. Ph.D. Dissertation, Universität des Saarlandes, Germany.

Krötzsch, M.; Rudolph, S.; and Hitzler, P. 2007a. Complexity of Horn description logics. In *Proc. 22nd AAAI Conf. (AAAI'07)*.

Krötzsch, M.; Rudolph, S.; and Hitzler, P. 2007b. 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.; and Cudré-Mauroux, P., eds., *Proc. 6th Int. Semantic Web Conference (ISWC 2007)*, volume 4825 of *LNCS*, 310–323. Springer.

Krötzsch, M.; Rudolph, S.; and Hitzler, P. 2008. Description logic rules. In *Proc. 18th European Conf. on Artificial Intelligence (ECAI-08)*, 80–84. IOS Press.

Motik, B.; Sattler, U.; and Studer, R. 2005. Query answering for OWL DL with rules. *J. of Web Semantics* 3:41–60.

Pérez-Urbina, H.; Motik, B.; and Horrocks, I. 2008a. Rewriting conjunctive queries over description logic knowledge bases. In *Proc. Int. Workshop on Semantics in Data and Knowledge Bases (SDKB-08)*.

Pérez-Urbina, H.; Motik, B.; and Horrocks, I. 2008b. Rewriting conjunctive queries under description logic constraints. In *Proc. Int. Workshop on Logic in Databases (LID-08)*.

Rosati, R. 2007. Conjunctive query answering in EL. In *Proc. 20th Int. Workshop on Description Logics (DL-07)*.

Rudolph, S.; Krötzsch, M.; and Hitzler, P. 2008. All elephants are bigger than all mice. In *Proc. 21st Int. Workshop on Description Logics (DL-08)*.