An efficient algorithm for reasoning over OWL EL ontologies with nominal schemas

David Carral, Pascal Hitzler, Joseph Zalewski

An Efficient Algorithm for Reasoning over OWL EL Ontologies with Nominal Schemas

David Carral · Pascal Hitzler · Joseph Zalewski

July 2021

Abstract Nominal schemas have been proposed as an extension to Description Logics (DL), the knowledge representation paradigm underlying the Web Ontology Language (OWL). They provide for a very tight integration of DL and rules. Nominal schemas can be understood as syntactic sugar on top of OWL. However, this naive perspective leads to inefficient reasoning procedures.

In order to develop an efficient reasoning procedure for the language $\mathcal{ELV}^{++}$, which results from extending the OWL profile language OWL EL with nominal schemas, we propose a transformation from $\mathcal{ELV}^{++}$ ontologies into Datalog-like rule programs that can be used for satisfiability checking and assertion retrieval. The use of this transformation enables the use of powerful Datalog engines to solve reasoning tasks over $\mathcal{ELV}^{++}$ ontologies. We implement and then evaluate our approach on several real-world, data-intensive ontologies, and find that it can outperform state-of-the-art reasoners such as Konclude and ELK. As a lesser side result we also provide a self-contained description of a rule-based algorithm for $\mathcal{EL}^{++}$ which does not require a normal form transformation.

The second and third author acknowledges funding by the National Science Foundation under award 2033521.

David Carral
LIRMM, Inria, University of Montpellier, CNRS
E-mail: david.carral@inria.fr

Joseph Zalewski
Data Semantics Laboratory, Kansas State University
Department of Computer Science, 1701D Platt St., Manhattan, KS 66506, USA
E-mail: jzalewski@ksu.edu

Pascal Hitzler
Data Semantics Laboratory, Kansas State University
Department of Computer Science, 1701D Platt St., Manhattan, KS 66506, USA
+1-785-532-6350, E-mail: hitzler@ksu.edu

Keywords Knowledge Representation · Reasoning · Description Logics · Datalog · OWL EL · Nominal Schemas

1 Introduction

Nominal schemas have been introduced in [29] based on preliminary ideas from [26,27]. Nominal schemas are essentially a syntactic extension to the Web Ontology Language (OWL) [13] which is based on Description Logics (DL) [2]. The primary purpose of nominal schemas is to allow DLs to behave more like *rules*, the primary alternative formalism to DLs in the ontology world. (We will fully introduce one standard rule language, Datalog, in this paper.) Nominal schemas can be understood as a generalization of the idea of DL-safe rules [15,25,33], a restricted type of rules permitting rule reasoning and description logic reasoning to be performed "simultaneously", for many standard DLs, without introducing undecidability. As such, nominal schemas make it possible to create OWL expressions equivalent to rules, covering a wide range of desirable kinds of rule, and therefore enable a very tight integration of the OWL and rule paradigms [23]. Essentially, nominal schemas are a variable version of *nominals*, or classes semantically constrained to have only one element, which is given an individual name; they are variables which can be bound to known individuals only, that is, individual names used in the knowledge base at hand. (This is convenient because there are finitely many of those.) For example, if a knowledge base contains 3 named individuals a, b, c, represented by nominals $\{a\}, \{b\}, \{c\}$, the axiom $(\exists R.\{x\} \sqsubseteq \exists S.\{x\})$ ($\{x\}$ is the nominal schema in this axiom) makes the assertion that whatever is related by R to a, b or c is also related to the same by S; however, it makes this assertion only regarding a, b, c. It does not say anything about items that are merely related by R to some thing. The following axiom, which is taken from [25] and is expressed in DL syntax, is a typical example for the use of nominal schemas.

$$\begin{aligned}&\exists \text{hasReviewAssignment}.(\{x\} \sqcap \exists \text{hasAuthor}.\{y\} \sqcap \exists \text{atVenue}.\{z\}) \sqcap \\ &\exists \text{hasSubmittedPaper}.(\exists \text{hasAuthor}.\{y\} \sqcap \exists \text{atVenue}.\{z\}) \\ &\sqsubseteq \exists \text{hasConflictingAssignedPaper}.\{x\}.\end{aligned}$$

In natural language, this axiom says: "Whoever has a review assignment x, with author y, at venue z, and has submitted a paper with author y at venue z, has conflicting assigned paper x", except of course it does not quite say that, it only asserts that this is true when the values of x, y, and z are individuals named somewhere in the knowledge base which has this axiom. One can think of the three nominal schemas $\{x\}$, $\{y\}$ and $\{z\}$ as mere placeholders for actual nominals. In fact, calling the number of named individuals in the knowledge base k, this axiom can be translated into k^3 axioms without nominal schemas by *fully grounding* the axiom, which is to say, creating k^3 new axioms by replacing the three nominal schemas by the k nominals in all possible combinations (see [25]). This translation perfectly preserves the semantics of the original axiom, and can be used as the intuition for its precise meaning,

as defined later. Full grounding eliminates nominal schemas and thus can be used, in principle, for reasoning over knowledge bases with nominal schemas. However, fully grounding an axiom with ℓ different nominal schemas results in the inclusion of k^ℓ new axioms without nominal schemas and the size of knowledge base often becomes unmanageable for current algorithms [8,9].

The motivation for introducing nominal schemas lies in bridging the gap between DL-based and rule-based approaches for ontology modelling [15,23, 25]. Often logical constraints which are easy to capture in one formalism are hard or impossible to capture in the other, and so it is considered desirable to have the "best of both worlds" by being able to systematically translate rules into DL axioms (or vice versa). Indeed, the example above arises from the rule

$$\begin{aligned}\text{hasReviewAssignment}(w,x) \wedge \text{hasAuthor}(x,y) \wedge \text{atVenue}(x,z) \wedge \\ \text{hasSubmittedPaper}(w,v) \wedge \text{hasAuthor}(v,y) \wedge \text{atVenue}(v,z) \\ \rightarrow \text{hasConflictingAssignedPaper}(w,x)\end{aligned}$$

if x, y and z are considered to be *DL-safe variables* [27]; that is, variables that bind only to named individuals present in the knowledge base and do not match to terms introduced during the reasoning process to satisfy existential restrictions.

1.1 Summary Of Related Work

In [29] it was shown that DL-safe binary Datalog is completely subsumed by nominal-schema-extended DL, and in [23] this was lifted to n-ary DL-safe Datalog. This means that nominal schemas allow for the incorporation of DL-safe SWRL [17,33] into the DL paradigm. It was also shown in [23] that the use of nominal schemas together with autoepistemic operators yields a DL which encompasses most of the major paradigms in non-monotonic logic programming and in local-closed-world-extended DL (see also [22]), thus constituting a major step towards establishing a unifying logic for major Semantic Web languages around the W3C standards OWL [13] and RIF [21].

It was shown in [29] that extending $\mathcal{SROIQ}$ with nominal schemas does not result in an increase of worst-case complexity, and it was also shown that a tractable fragment can be obtained which encompasses both OWL EL and the DL-safe version of OWL RL [13] (and, more generally, Datalog under the Herbrand semantics provided there is a bound on the number of variables per rule). The complexities of different DL languages extended with nominal schemas are studied in [30]. However, despite this, first attempts to arrive at an efficient reasoning algorithm with nominal schemas have had limited success: [24] reported on a corresponding extension of tableaux algorithms, while [42] reported on a resolution-based algorithm for the tractable fragment—but neither of these algorithms looked promising enough in terms of scalability to even attempt an implementation. However, an adaptation of the reasoning optimization technique known as *absorption* has led to a significantly improved

reasoning algorithm with nominal-schema-extended $\mathcal{SROIQ}$ [37,38]; that is, the logic underlying the OWL DL language.

1.2 Contributions

In this paper, we present an efficient approach to solve assertion retrieval and satisfiability checking over OWL EL ontologies with nominal schemas. More precisely, we present a transformation from OWL EL ontologies into Datalog programs that preserves assertion entailment. Using this transformation, we can solve assertion retrieval over OWL EL ontologies using the following two-step approach.

1. Given an ontology $\mathcal{O}$, we compute the Datalog program $\mathcal{P}_\mathcal{O}$ using the transformation presented in Section 4. Note that, for every assertion α, we have that $\mathcal{O} \models \alpha$ if and only if $\mathcal{P}_\mathcal{O}$ entails a translation of α (see Theorem 2).
2. We use an efficient Datalog engine to compute all of the inferences of $\mathcal{P}_\mathcal{O}$, which allows us to obtain all the assertions entailed by $\mathcal{O}$.

Furthermore, there is a particular fact which is derivable from $\mathcal{P}_\mathcal{O}$ iff $\mathcal{O}$ is unsatisfiable.

We also report on an implementation of the above two-step approach, which makes use of VLog [40,41] as a Datalog engine; and on a corresponding experimental evaluation, which shows that our approach is indeed rather efficient. Namely, we compare the performance of our approach on several real-world ontologies and find that our implementation can outperform the DL reasoner Konclude [39]—considered as one of the leading DL reasoners [35]—and the OWL EL reasoner ELK [20].

In summary, our main contributions are the following.

- We introduce and prove correctness of a worst-case reasoning procedure for $\mathcal{ELV}^{++}$ ontologies based on a translation into Datalog.
- We implement a prototype based on this algorithm.
- We conduct an empirical evaluation that shows performance gains over state-of-the-art DL reasoners.
- We introduce a rule-based reasoning algorithm for $\mathcal{EL}^{++}$ not using normal form transformation.

The paper is structured as follows.

- Section 2: we introduce the DL languages $\mathcal{EL}^{++}$ and $\mathcal{ELV}^{++}$, and Datalog.
- Section 3: we present a translation from normalised ontologies without nominal schemas into Datalog programs and show that this translation preserves assertion entailment.
- Section 4.1: we introduce grounding [29], which is a naive reasoning procedure for nominal schemas.
- Section 4.2: leveraging results from Sections 3 and 4.1, we define a translation from (possibly not normalised) ontologies with nominal schemas into Datalog programs that preserves assertion entailment.

– Section 5: we comment on an evaluation of our reasoning approach.
– Section 6: we elaborate about further work.

This paper is a very significantly extended and revised version of [9].

2 Preliminaries

2.1 Description Logics

We summarize basic notions from DL used in this paper. We assume some familiarity with the topic and otherwise refer the reader to the literature for further details: for a thorough theoretical introduction see [2]; an extended introduction to DL and Semantic Web technologies and standards is provided in [14], where also the relationships between DL and the different OWL standards are explained in detail.

We formally introduce the DL languages $\mathcal{EL}^{++}$ [3] and $\mathcal{ELV}^{++}$ [29]. Although the focus of this paper is on reasoning over $\mathcal{ELV}^{++}$, we make use of the language $\mathcal{EL}^{++}$ in some of our formal arguments.

Let an $\mathcal{ELV}^{++}$ *signature* be a tuple ($\mathbf{C}$, $\mathbf{R}$, $\mathbf{I}$, $\mathbf{N}$) where $\mathbf{C}$, $\mathbf{R}$, $\mathbf{I}$, $\mathbf{N}$ are pairwise disjoint countably infinite sets. We will call the elements of $\mathbf{C}$ *concept names*, and likewise those of $\mathbf{R}$ *roles*, $\mathbf{I}$ *individuals* and $\mathbf{N}$ *nominal variables*, respectively. Let $\mathbf{C}$ contain two distinguished elements $\top$ and $\bot$. Next we define expressions and axioms over a signature - from now on, the signature may not be explicitly mentioned. The set $\mathbf{E}$ of $\mathcal{ELV}^{++}$ *concept expressions* (or simply *expressions*) over ($\mathbf{C}$, $\mathbf{R}$, $\mathbf{I}$, $\mathbf{N}$) is defined by the following grammar

$$\mathbf{E} ::= (\mathbf{E} \sqcap \mathbf{E}) \mid \exists R.\mathbf{E} \mid \exists R.\mathsf{Self} \mid \mathsf{Dom}(R) \mid \mathsf{Ran}(R) \mid C \mid \{a\} \mid \{x\}$$

where $C \in \mathbf{C}$, $R \in \mathbf{R}$, $a \in \mathbf{I}$, and $x \in \mathbf{N}$. Let $C_1 \sqcap \ldots \sqcap C_k$ be an abbreviation for $C_1 \sqcap (C_2 \sqcap (\ldots C_k) \ldots)$. Note the inclusion of *nominal schemas* in $\mathbf{E}$; that is, concept expressions of the form $\{x\}$ with $x \in \mathbf{N}$.

An *($\mathcal{ELV}^{++}$) terminological axiom* is a syntactic expression given by the grammar

$$\mathbf{TA} ::= \mathbf{E} \sqsubseteq \mathbf{E} \mid \mathbf{E} \equiv \mathbf{E} \mid R_1 \circ \ldots \circ R_n \sqsubseteq S \mid R \equiv S \mid \mathsf{Ref}(R) \mid \mathsf{Tran}(R)$$

with $R_i, R, S \in \mathbf{R}$. An *($\mathcal{ELV}^{++}$) assertion* (or *assertional axiom*) is a syntactic expression given by

$$\mathbf{AA} ::= \mathbf{E}(a) \mid \neg\mathbf{E}(a) \mid R(a,b) \mid \neg R(a,b) \mid a \approx b \mid a \not\approx b$$

where $R \in \mathbf{R}, a, b \in \mathbf{I}$. An *($\mathcal{ELV}^{++}$) axiom* is a terminological axiom or an assertion. See Table 1 for reference.

An $\mathcal{EL}^{++}$ *axiom* is an axiom without occurrences of nominal variables.

Consider a set $\mathcal{S}$ of $\mathcal{ELV}^{++}$ axioms. Then, we define $\sqsubseteq^*_{\mathcal{S}}$ as the minimal transitive and reflexive relation over roles such that $R \sqsubseteq^*_{\mathcal{S}} S$ if $R \sqsubseteq S$, $R \equiv S$, or $S \equiv R$ are in $\mathcal{S}$. A role R is *simple with respect to* $\mathcal{S}$ iff, for every S where

Concept Expression	**Syntax**	**Semantics**
Primitive Concept Name	A	$A^{\mathcal{I}}$
Conjunction	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
Existential Restriction	$\exists R.C$	$\{d \mid \exists e[(d,e) \in R^{\mathcal{I}}, e \in C^{\mathcal{I}}]\}$
Self Restriction	$\exists R.\mathsf{Self}$	$\{d \mid (d,d) \in R^{\mathcal{I}}\}$
Domain Restriction*	$\mathsf{Dom}(R)$	$\{d \mid (d,e) \in R^{\mathcal{I}}\}$
Range Restriction	$\mathsf{Ran}(R)$	$\{d \mid (e,d) \in R^{\mathcal{I}}\}$
Nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$
Nominal Schema	$\{x\}$	$\{\mathcal{Z}(x)^{\mathcal{I}}\}$
Top	$\top$	$\Delta^{\mathcal{I}}$
Bottom	$\bot$	$\emptyset$

Axiom	**Syntax**	**Semantics**
Concept Inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
Concept Equivalence*	$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$
Role Inclusion	$R_1 \circ \ldots \circ R_n \sqsubseteq S$	$(R_1 \circ \ldots \circ R_n)^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
Role Equivalence*	$R \equiv S$	$R^{\mathcal{I}} = S^{\mathcal{I}}$
Reflexivity*	$\mathsf{Ref}(R)$	$\{(t,t) \mid t \in \Delta^{\mathcal{I}}\} \subseteq R^{\mathcal{I}}$
Transitivity*	$\mathsf{Tran}(R)$	$(R \circ R)^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
Concept Assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
Negative C. Assertion	$\neg C(a)$	$a^{\mathcal{I}} \notin C^{\mathcal{I}}$
Role Assertion	$R(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$
Negative R. Assertion	$\neg R(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin R^{\mathcal{I}}$
Equality Assertion	$a \approx b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$
Inequality Assertion	$a \not\approx b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$

Table 1 $\mathcal{ELV}^{++}$ Syntax and Semantics. $A \in \mathbf{C} \setminus \{\top, \bot\}, C_{(i)}, D \in \mathbf{E}$, $R_{(i)}, S \in \mathbf{R}$, $a, b \in \mathbf{I}$, and $x \in \mathbf{N}$.

* marks redundant constructs.

$S \sqsubseteq^*_{\mathcal{S}} R$, there are no axioms in $\mathcal{S}$ of the form (i) $\mathsf{Tran}(S)$ or (ii) $S_1 \circ \ldots \circ S_k \sqsubseteq S$ with $k > 1$.

An $\mathcal{ELV}^{++}$ *ontology* $\mathcal{O}$ is a set of $\mathcal{ELV}^{++}$ axioms that additionally satisfies all the following conditions.

1. For all $R \in \mathbf{R}$, if $\mathsf{Ref}(R) \in \mathcal{O}$ or the expression $\exists R.\mathsf{Self}$ occurs in $\mathcal{O}$, then R is simple with respect to $\mathcal{O}$.
2. If $R_1 \circ \ldots \circ R_k \sqsubseteq S \in \mathcal{O}$, $k > 1$, and $S \sqsubseteq^*_{\mathcal{O}} R$, and $\mathsf{Ran}(R) \sqsubseteq B \in \mathcal{O}$, then $\mathsf{Ran}(R_k) \sqsubseteq B \in \mathcal{O}$.

An $\mathcal{EL}^{++}$ *ontology* $\mathcal{O}$ is an $\mathcal{ELV}^{++}$ ontology containing only $\mathcal{EL}^{++}$ axioms; that is, an $\mathcal{ELV}^{++}$ ontology without occurrences of nominal schemas.

Condition (1) is considered in existing definitions of the $\mathcal{EL}^{++}$ language [28] and is in place to preserve decidability. An undecidability proof for $\mathcal{EL}^{++}$ without this restriction can be easily produced along the lines of the proofs presented in Section 2.5.1 of [19]. Dropping (2) was shown to lead to undecidability in [4], which is also where this condition was initially introduced.

We do not consider any structural restrictions regarding role regularity [18] in our definition of $\mathcal{ELV}^{++}$ even though some of these apply to OWL EL

[32]. These restrictions, which are in place to make OWL EL a strict subset of OWL DL in terms of expressivity, are not necessary for preserving termination or tractability of $\mathcal{ELV}^{++}$ reasoning algorithms. Consequently, they are not considered in this publication.

Let the *size* of a concept (resp. an axiom, an ontology) be the total number of symbols in it.

The semantics of an $\mathcal{ELV}^{++}$ ontology $\mathcal{O}$ is given through the definition of *interpretations* and *nominal variable assignments*. An interpretation $\mathcal{I}$ is a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is a set and $\cdot^{\mathcal{I}}$ is a function that maps each individual occurring in an axiom in the ontology to an element of $\Delta^{\mathcal{I}}$, and likewise each concept name in the ontology to a subset of $\Delta^{\mathcal{I}}$, and each role name in the ontology to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ (a binary relation). Also, an interpretation must have $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $\bot^{\mathcal{I}} = \emptyset$. This is shown in Table 1 for convenience. A nominal variable assignment $\mathcal{Z}$ is a function $\mathcal{Z} : \mathbf{N} \rightarrow \mathbf{I}_{\mathcal{O}}$, where $\mathbf{I}_{\mathcal{O}}$ is the set of individual names appearing in $\mathcal{O}$. Given $\mathcal{Z}$, an interpretation $\mathcal{I}$ can be extended in a canonical way to map all concept expressions over the individual, class and role names used in the ontology to subsets of $\Delta^{\mathcal{I}}$. This extension with respect to $\mathcal{Z}$ is defined recursively. $\{x\}^{\mathcal{I}}$ is defined as $\{\mathcal{Z}(x)^{\mathcal{I}}\}$ for any nominal-schema expression $\{x\}$ occurring in $\mathcal{O}$, and otherwise the extension of $\mathcal{I}$ is made in the usual way per description logic conventions: $\top^{\mathcal{I}}$ is defined to be $\Delta^{\mathcal{I}}$, $\bot^{\mathcal{I}}$ to be the empty set, $(C_1 \sqcap C_2)^{\mathcal{I}}$ to be $C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$, and so on; see Table 1 for reference. We will sometimes write the extension of $\mathcal{I}$ with respect to $\mathcal{Z}$ as just $\mathcal{I}$, when it is clear from context which Z is intended.

An interpretation $\mathcal{I}$ and a nominal variable assignment $\mathcal{Z}$ *satisfy* an $\mathcal{ELV}^{++}$ axiom α, written $\mathcal{I}, \mathcal{Z} \models \alpha$, if the corresponding condition shown in the lower part of Table 1 holds. For instance, $\mathcal{I}, \mathcal{Z} \models C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. An interpretation $\mathcal{I}$ *satisfies* α, written $\mathcal{I} \models \alpha$, if $\mathcal{I}, \mathcal{Z} \models \alpha$ for all nominal variable assignments $\mathcal{Z}$. For $\mathcal{EL}^{++}$ axioms α, obviously $\mathcal{I}, \mathcal{Z} \models \alpha$ for all $\mathcal{Z}$ iff $\mathcal{I}, \mathcal{Z} \models \alpha$ for some $\mathcal{Z}$, since $\mathcal{EL}^{++}$ concepts do not contain nominal variables. So we will write $\mathcal{I} \models \alpha$ for $\mathcal{EL}^{++}$ axioms with the understanding that $\mathcal{Z}$ is irrelevant.

An interpretation $\mathcal{I}$ *satisfies* an ontology $\mathcal{O}$, written $\mathcal{I} \models \mathcal{O}$, if $\mathcal{I} \models \alpha$ for every $\alpha \in \mathcal{O}$. If this is the case, we say that $\mathcal{I}$ is a *model* of $\mathcal{O}$. An ontology $\mathcal{O}$ is *satisfiable* if and only if there exists some model for $\mathcal{O}$. An ontology $\mathcal{O}$ *entails* an axiom α, written $\mathcal{O} \models \alpha$, if $\mathcal{I} \models \alpha$ for every model $\mathcal{I}$ of $\mathcal{O}$. Note: Many of the axiom constructs in Table 1 are redundant; they can be considered aliases for some set of other axioms combined with expression constructors, because the axiom and its corresponding set of axioms entail each other (or, they are *logically equivalent*). These are included only for convenience, so the reader versed in description logic (or the OWL standard, in which all of them are indeed primitives) can quickly see which of the standard DL axiom types are expressible in $\mathcal{ELV}^{++}$. The redundant constructs (marked with asterisks in Table 1) are: Concept Equivalence - $C \equiv D$ is equivalent to $\{C \sqsubseteq D, D \sqsubseteq C\}$, Role Equivalence - $R \equiv S$ is equivalent to $\{R \sqsubseteq S, S \sqsubseteq R\}$, Reflexivity - $\mathsf{Ref}(R)$ is equivalent to $\exists R.\mathsf{Self} \equiv \top$, Transitivity - $Tran(R)$ is equivalent to $R \circ R \sqsubseteq R$. The *concept* expression $\mathsf{Dom}(R)$ is also redundant; $\mathsf{Dom}(R)^{\mathcal{I}} = (\exists R.\top)^{\mathcal{I}}$ for every $\mathcal{I}, \mathcal{Z}$.

It has been shown in [30] that description logics with nominal schemas behave quite similarly if nominal variable assignments are allowed to take values in the infinite set $\mathbf{I}$ rather than $\mathbf{I}_{\mathcal{O}}$, but we use the more classical semantics.

The main reasoning task in DL is *satisfiability checking*; that is, checking if there exists at least one model for a given ontology. Besides satisfiability checking, we focus on solving *assertion retrieval*; that is, computing *all* of the assertions that follow from an ontology.

2.2 Datalog

Our reasoning algorithm for $\mathcal{ELV}^{++}$ is based on the rule language Datalog extended with function symbols, which is briefly introduced in this section. Moreover, we introduce the *chase*, a bottom up materialisation procedure that can be used to compute all of the facts that follow from a Datalog theory. As with DL, we assume that the reader is familiar with the topic, and otherwise we refer her to the literature [1].

Let a *Datalog signature* be a tuple ($\mathbf{P}$, $\mathbf{F}$, $\mathbf{V}$) where $\mathbf{P}$, $\mathbf{F}$, $\mathbf{V}$ are pairwise disjoint countably infinite sets (call the elements of $\mathbf{P}$ *predicates*, likewise $\mathbf{F}$ *function symbols* and $\mathbf{V}$ *variables*) together with an assignment of some non-negative integer *arity*, $\mathsf{ar}(s)$, for each symbol $s \in \mathbf{P} \cup \mathbf{F}$, and a distinguished element $\approx \in \mathbf{P}$ with $\mathsf{ar}(\approx) = 2$. The set of *constants* $\mathbf{O}$ (over a given Datalog signature; again we may omit explicit mention of signatures later) is the set of all function symbols of arity 0. We inductively define the set of *terms* $\mathbf{T}$ as follows: $\mathbf{O} \cup \mathbf{V} \subseteq \mathbf{T}$ and, for every function symbol f with $\mathsf{ar}(f) \geq 1$ and every sequence of terms $t_1, \ldots, t_{\mathsf{ar}(f)}$, we have that the syntactic expression $f(t_1, \ldots, t_{\mathsf{ar}(f)}) \in \mathbf{T}$. An *atom* is an expression $p(t_1, \ldots, t_k)$ with $p \in \mathbf{P}$, $\mathsf{ar}(p) = k$ and $t_i \in \mathbf{T}$ for all $i = 1, \ldots, k$. As is customary, we slightly abuse notation and write $t \approx u$ instead of $\approx(t, u)$ for atoms defined over the special equality predicate $\approx$.

A *rule* is a formula of the form $B \to H$ where B and H are conjunctions of atoms with $H \neq \emptyset$. We refer to B and H as the *body* and the *head* of the rule, respectively. We often identify conjunctions of atoms, such as B and H, with sets. Note that we have adopted a rather liberal definition of Datalog: we allow for multiple atoms in the head of a rule, function symbols, and variables occurring in the head that do not occur in the body. However, our rules are not *existential rules*; variables occurring only in the head are to be read as "universally quantified", as implied by Definition 1 below. A *program* is a set of rules. A *ground rule* (respectively, ground atom, ground term) is a rule (atom, term) in which no variables appear. A *fact* is a ground rule with an empty body. As is customary, we simply write $p(c_1, \ldots, c_k)$ instead of $\to p(c_1, \ldots, c_k)$ when dealing with facts.

A *substitution* is a partial function mapping variables to terms. The application of a substitution σ to a formula φ of atoms, written $\varphi\sigma$, is the formula that results from replacing all syntactic occurrences of every variable x in the domain of σ with $\sigma(x)$ if the latter is defined. For instance,

given the substitution $\sigma = \{(x, a)\}$ and the atom $A(x, f(x))$, we have that $A(x, f(x))\sigma = A(a, f(a))$. Call a substitution *ground* if all its values are ground terms.

Definition 1 Consider a rule $\rho = B \to H$, a set of facts $\mathcal{F}$, and a program $\mathcal{P}$. Then, let $\rho(\mathcal{F})$ be the fact set containing $H\sigma$ for every ground substitution σ such that σ is defined on all variables in ρ and $B\sigma \subseteq \mathcal{F}$, and let $\mathcal{P}(\mathcal{F}) = \bigcup_{\rho\in\mathcal{P}} \rho(\mathcal{F})$. The *chase sequence of* $\mathcal{P}$ is the sequence $\mathcal{P}^0, \mathcal{P}^1, \ldots$ of fact sets such that $\mathcal{P}^0 = \emptyset$ and $\mathcal{P}^i = \mathcal{P}(\mathcal{P}^{i-1})$ for all $i \geq 1$. The *chase of* $\mathcal{P}$ is the set $\mathsf{Ch}(\mathcal{P}) = \bigcup_{i\geq 0} \mathcal{P}^i$. We say that the chase of $\mathcal{P}$ *terminates* if $\mathsf{Ch}(\mathcal{P})$ is finite; that is, if there is some $i \geq 0$ with $\mathcal{P}^j = \mathcal{P}^i$ for all $j \geq i$.

Note that, since we consider Datalog programs with function symbols, it is indeed possible that the chase does not terminate because an infinite number of terms is introduced during its computation.

Rules (resp. atoms, conjunctions of atoms) r and s are *isomorphic* if there exists a bijective substitution σ with $r\sigma = s$. Let the *size* of an atom (resp., a rule, a program) be the total number of symbols in it.

In subsequent sections we define mappings from ontologies to Datalog programs and show that that the latter can be used to solve reasoning tasks over the former. To appropriately define these ontology-to-program mappings, let ($\mathbf{C}$,$\mathbf{R}$,$\mathbf{I}$,$\mathbf{N}$) be an $\mathcal{ELV}^{++}$ signature; we define a corresponding Datalog signature ($\mathbf{P}$,$\mathbf{F}$,$\mathbf{V}$). $\mathbf{P} = \mathbf{C} \cup \mathbf{R} \cup \{\mathsf{Named}\}$, where Named is a symbol not occurring in ($\mathbf{C}$,$\mathbf{R}$,$\mathbf{I}$,$\mathbf{N}$). Let $\star$ be another symbol not occurring in our $\mathcal{ELV}^{++}$ signature. For a $C \in \mathbf{E}$, we define $C_\star$ as the concept expression that results from replacing every syntactic occurrence of any individual or nominal variable in C by $\star$. For each $R \in \mathbf{R}, C \in \mathbf{E}$, let $f_{RC_\star}$ be a fresh function symbol unique for R and $C_\star$, with arity equal to the number of (not necessarily distinct) individuals and nominal variables occurring in C. Now $\mathbf{F} = \mathbf{I} \cup \{f_{RC_\star} | R \in \mathbf{R}, C \in \mathbf{C}\}$.

Example 1 Let C be the description logic expression $\exists R.(\{u\} \sqcap \exists S.(\{u\} \sqcap \{v\}))$. $C_\star$ is the term $\exists R.(\{\star\} \sqcap \exists S.(\{\star\} \sqcap \{\star\}))$.
For each role T, our Datalog signature will include a fresh function symbol $f_{T[\exists R.(\{\star\}\sqcap\exists S.(\{\star\}\sqcap\{\star\}))]}$, which has arity 3.

Definition 2 Given $\exists R.C \in \mathbf{E}$ with $R \in \mathbf{R}$ and $C \in \mathbf{E}$, let $\mathsf{f}(\exists R.C) = f_{RC_\star}(t_1, \ldots, t_k)$, where $t_1, \ldots, t_k$ is the sequence of all (not necessarily distinct) individuals and nominal variables occurring in C sorted by the order in which they syntactically appear.

Example 2 Reusing the expression from Example 1, $\mathsf{f}(\exists R.(\{u\} \sqcap \exists S.(\{u\} \sqcap \{v\}))) = f_{R[(\{\star\}\sqcap\exists S.(\{\star\}\sqcap\{\star\}))]}(u, u, v)$.

Example 3 If C is an expression with no nominal variables, like $\exists R.(A \sqcap \{a\})$ for individual name a, $\mathsf{f}(C)$ will be a ground term:
$\mathsf{f}(C) = f_{R[A\sqcap\{\star\}]}(a)$.

For a while, we will be dealing with $\mathsf{f}(\exists R.C)$ only for classes C containing no nominal variables, and in this case we will write the ground term $\mathsf{f}(\exists R.C)$ as f_{RB}. We may informally refer to it as a "constant." $\mathbf{V} = \mathbf{N} \cup \mathbf{V'}$, where $\mathbf{V'}$ is an infinite set disjoint from all others mentioned so far, which will be used to obtain "fresh" variables during translation into Datalog.

Let $\mathsf{ar}(C) = 1$ for all $C \in \mathbf{C}$, $\mathsf{ar}(R) = 2$ for all $R \in \mathbf{R}$, $ar(\mathsf{Named}) = 1$, $ar(\top) = 1$, $ar(\bot) = 0$, $ar(a) = 0$ for $a \in \mathbf{I}$, and $ar(f_{RC})$ be the number of occurrences of nominals and nominal schemas in C.

The function symbols f_{RC} are used to introduce distinct functional terms during the reasoning process to satisfy existential restrictions enforced by concept expressions of the form $\exists R.C$ with C a concept expression containing nominal schemas.

$\bot$ is made 0-ary because it will be used as a "failure signal", which indicates unsatisfiability if it can be deduced; it does not matter which specific elements are deduced to be in $\bot$.

3 A Reasoning Algorithm for Normalized $\mathcal{EL}^{++}$ Ontologies

In this section, we define a normal form for $\mathcal{EL}^{++}$ ontologies and a Datalog-based algorithm to reason over normalized $\mathcal{EL}^{++}$ ontologies. This preliminary algorithm is used in later sections to prove correctness of other reasoning procedures which do not require a preliminary normalization step.

The reasoning algorithm presented in this section is similar to the one in [28]; the main difference being that, in [28], a fixed set of rules is used for reasoning, while $\mathcal{EL}^{++}$ axioms are translated into facts. A fixed set of rules is sufficient in this case because $\mathcal{EL}^{++}$ can be cast into a very limited normal form; however, we did not find a similarly limited normal form for $\mathcal{ELV}^{++}$. Our final reasoning algorithm for $\mathcal{EL}^{++}$, as presented in this section, disposes of the need for such a normal form by providing a translation of TBox axioms into *rules* and ABox axioms into *facts*. This will give us the additional flexibility to extend the algorithm to $\mathcal{ELV}^{++}$ in subsequent sections of this paper.

Definition 3 An $\mathcal{EL}^{++}$ ontology is in *normal form* if it only contains axioms in the forms, $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B, \exists R.A \sqsubseteq B, \exists R.\mathsf{Self} \sqsubseteq B, \mathsf{Ran}(R) \sqsubseteq B, \top \sqsubseteq B, A \sqsubseteq \exists R.B, A \sqsubseteq \exists R.\mathsf{Self}, A \sqsubseteq \{a\}, A \sqsubseteq \bot, R \sqsubseteq S, R_1 \circ \ldots \circ R_k \sqsubseteq S, A(a), \neg A(a), R(a,b), \neg R(a,b), a \approx b, a \not\approx b$; that is, those appearing on the left side of figure 1.

Various normal form transformations for $\mathcal{EL}^{++}$ exist in the literature, for example a satisfiability-perserving one in [28]. Our normal form transformation is a conservative extension:

Proposition 1 *For an $\mathcal{EL}^{++}$ ontology $\mathcal{O}$, there is another $\mathcal{EL}^{++}$ ontology $\mathcal{O}'$ in normal form such that $\mathcal{O} \models \alpha$ if and only if $\mathcal{O}' \models \alpha$ for every axiom α containing only predicates and constants occurring in $\mathcal{O}$.*

$$
\begin{aligned}
A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B &\mapsto \{A_1(x) \wedge \ldots \wedge A_n(x) \rightarrow B(x)\} && (1)\\
\exists R.A \sqsubseteq B &\mapsto \{R(x,y) \wedge A(y) \rightarrow B(x)\} && (2)\\
\exists R.\mathsf{Self} \sqsubseteq B &\mapsto \{R_{\mathsf{Self}}(x) \rightarrow B(x)\} && (3)\\
\mathsf{Ran}(R) \sqsubseteq B &\mapsto \{R(y,x) \rightarrow B(x)\} && (4)\\
\top \sqsubseteq B &\mapsto \{\top(x) \rightarrow B(x)\} && (5)\\
A \sqsubseteq \exists R.B &\mapsto \{A(x) \rightarrow R(x, f_{RB}) \wedge B(f_{RB})\} && (6)\\
A \sqsubseteq \exists R.\mathsf{Self} &\mapsto \{A(x) \rightarrow R_{\mathsf{Self}}(x)\} && (7)\\
A \sqsubseteq \{a\} &\mapsto \{A(x) \rightarrow a \approx x\} && (8)\\
A \sqsubseteq \bot &\mapsto \{A(x) \rightarrow \bot\} && (9)\\
R \sqsubseteq S &\mapsto \{R(x,y) \rightarrow S(x,y), R_{\mathsf{Self}}(x) \rightarrow S_{\mathsf{Self}}(x)\} && (10)\\
R_1 \circ \ldots \circ R_k \sqsubseteq S &\mapsto \{R_1(x_0,x_1) \wedge \ldots \wedge R_k(x_{k-1},x_k) \rightarrow S(x_0,x_k)\} && (11)\\
A(a) &\mapsto \{A(a)\} && (12)\\
\neg A(a) &\mapsto \{A(a) \rightarrow \bot\} && (13)\\
R(a,b) &\mapsto \{R(a,b)\} && (14)\\
\neg R(a,b) &\mapsto \{R(a,b) \rightarrow \bot\} && (15)\\
a \approx b &\mapsto \{a \approx b\} && (16)\\
a \not\approx b &\mapsto \{a \approx b \rightarrow \bot\} && (17)
\end{aligned}
$$

Listing 1 Function δ. In the above, $A_{(i)}, B \in \mathbf{C} \setminus \{\top, \bot\}$, $R_{(i)}, S \in \mathbf{R}$, $a, b \in \mathbf{I}$, and $k > 1$.

Proof Given an $\mathcal{EL}^{++}$ ontology $\mathcal{O}$, let $\mathcal{O}_1, \ldots, \mathcal{O}_k$ be a sequence of ontologies such that $\mathcal{O}_1 = \mathcal{O}$ and, for every $i \geq 1$, $\mathcal{O}_{i+1}$ is the ontology that results from replacing every axiom $\alpha \in \mathcal{O}_i$ not in normal form by the axioms in $\mathsf{NF}(\alpha)$ with NF the function from Listing 2. It is clear that only a linear number of such replacements is necessary to arrive at $\mathcal{O}_k$, which is an $\mathcal{EL}^{++}$ ontology in normal form. This is indeed the case, as none of the restrictions defined for $\mathcal{EL}^{++}$ ontologies is violated during the construction of the sequence $\mathcal{O}_1, \ldots, \mathcal{O}_k$. In particular, note the use of the fresh concept name X_D when normalizing axioms of the form $\mathsf{Ran}(R) \sqsubseteq B$.

For every pair of ontologies $\mathcal{O}_i$ and $\mathcal{O}_{i+1}$ in the sequence the relation observed in Proposition 1 is true: any interpretation satisfying $\mathcal{O}_i$ can be extended to an interpretation of $\mathcal{O}_{i+1}$ by interpreting every (possibly introduced) fresh concept name as the least set of domain elements to satisfy every axiom in $\mathcal{O}_{i+1}$ (using the original interpretation for all other symbols). Conversely, any interpretation that satisfies $\mathcal{O}_{i+1}$ necessarily satisfies $\mathcal{O}_i$. Since these observations are easily verified for each pair of ontologies $\mathcal{O}_i$ and $\mathcal{O}_{i+1}$ in the sequence, the proposition follows by induction. □

Hence, given some $\mathcal{EL}^{++}$ ontology $\mathcal{O}$, there is always some normalized ontology $\mathcal{O}'$ that can alternatively be used to solve reasoning tasks over $\mathcal{O}$.

$$
\begin{aligned}
C \equiv D &\mapsto \{C \sqsubseteq D, D \sqsubseteq C\} && (1)\\
\mathsf{dom}(R) \sqsubseteq D &\mapsto \{\exists R.\top \sqsubseteq D\} && (2)\\
\mathsf{ran}(R) \sqsubseteq D &\mapsto \{\mathsf{ran}(R) \sqsubseteq X_D, X_D \sqsubseteq D\} && (3)\\
R \equiv S &\mapsto \{R \sqsubseteq S, S \sqsubseteq R\} && (4)\\
\mathsf{Ref}(R) &\mapsto \{\top \sqsubseteq \exists R.\mathsf{Self}\} && (5)\\
\mathsf{Trans}(R) &\mapsto \{R \circ R \sqsubseteq R\} && (6)\\
E \sqsubseteq F &\mapsto \{E \sqsubseteq X, X \sqsubseteq F\} && (7)\\
A \sqsubseteq D_1 \sqcap \ldots \sqcap D_n &\mapsto \{A \sqsubseteq D_1, \ldots, A \sqsubseteq D_n\} && (8)\\
A \sqsubseteq \exists R.D &\mapsto \{A \sqsubseteq \exists R.X, X \sqsubseteq D\} && (9)\\
A \sqsubseteq \top &\mapsto \emptyset && (10)\\
C_1 \sqcap \ldots \sqcap C_k \sqsubseteq B &\mapsto \{C_1 \sqsubseteq X_1, \ldots, C_k \sqsubseteq X_k, X_1 \sqcap \ldots \sqcap X_k \sqsubseteq B\} && (11)\\
\exists R.C \sqsubseteq B &\mapsto \{C \sqsubseteq X, \exists R.X \sqsubseteq B\} && (12)\\
\bot \sqsubseteq B &\mapsto \emptyset && (13)\\
\{a\} \sqsubseteq B &\mapsto \{B(a)\} && (14)\\
C(a) &\mapsto \{X(a), X \sqsubseteq C\} && (15)\\
\neg C(a) &\mapsto \{C \sqsubseteq X, \neg X(a)\} && (16)
\end{aligned}
$$

Listing 2 Function NF. In the above, $A, B \in \mathbf{C} \setminus \{\top, \bot\}$, $C_{(i)}, D_{(j)} \in \mathbf{E}$, $E, F \in (\mathbf{E} \setminus \mathbf{C})$, all $X_{(i)}$ are fresh concept names, and X_D is a fresh concept name unique for D.

3.1 The Translation

An explanation is in order before we present the form of our rule transformation. It may appear that the rules we derive from an ontology $\mathcal{O}$ are meant to axiomatize the behavior of elements in a model of $\mathcal{O}$. For instance, an axiom $A \sqsubseteq B$ becomes a rule $A(x) \rightarrow B(x)$, and indeed a model M of this axiom, treated as a set of facts, is closed under the chase of the rule $A^M(x) \rightarrow B^M(x)$ (the rule is "true"). Similar considerations seem to explain most parts of our translation. However, this appearance is deceptive. Our true intention is that Datalog constants represent small subsets of a model of $\mathcal{O}$, the individual-name constants being single-point subsets, but other constants (f_{RB}) potentially having several elements. A Datalog atom $A(t)$ is to be read as "t is a subset of A^M", and $R(t, u)$ as "u is a subset of the range of R, and contains all R-successors of t"; $R_{\mathsf{Self}}(t)$ is read "t is a subset of $(\exists R.\mathsf{Self})^M$". This kind of translation leads to a very direct (although heavy with trivial inductive cases) proof of the soundness direction of the main Theorem 1 below. The above should all be taken as intuition; now we will define the translation purely by cases. However, the intuition will return in a precise way in the proof just mentioned.

Definition 4 Given an $\mathcal{ELV}^{++}$ ontology $\mathcal{O}$, let $\mathsf{Aux}_{\mathcal{O}}$ be the union of the sets of rules $\mathsf{Eq}_{\mathcal{O}}$, $\mathsf{Top}_{\mathcal{O}}$, $\mathsf{Self}_{\mathcal{O}}$, $\mathsf{Named}_{\mathcal{O}}$, and $\mathsf{Bottom}_{\mathcal{O}}$ described in Listing 3.

The sets of rules $\mathsf{Eq}_{\mathcal{O}}$, $\mathsf{Top}_{\mathcal{O}}$, and $\mathsf{Self}_{\mathcal{O}}$ are used to axiomatize the meaning of the equality predicate, the top predicate and the self constructor, respec-

$$
\begin{aligned}
\mathsf{Eq}_{\mathcal{O}} = {} & \{x \approx y \to y \approx x, x \approx y \wedge y \approx z \to x \approx z\} \cup \\
& \{A(x) \wedge x \approx y \to A(y) \mid A \in \mathbf{C}_{\mathcal{O}} \setminus \{\bot\}\} \cup \\
& \{R(x,y) \wedge x \approx z \to R(z,y), R(x,y) \wedge y \approx z \to R(x,z) \mid R \in \mathbf{R}_{\mathcal{O}}\} \cup \\
& \{\top(x) \to x \approx x\} \\
\mathsf{Top}_{\mathcal{O}} = {} & \{A(x) \to \top(x) \mid A \in \mathbf{C}_{\mathcal{O}}\} \cup \{R(x,y) \to \top(x) \wedge \top(y) \mid R \in \mathbf{R}_{\mathcal{O}}\} \\
\mathsf{Self}_{\mathcal{O}} = {} & \{R(x,x) \wedge \mathsf{Named}(x) \to R_{\mathsf{Self}}(x), R_{\mathsf{Self}}(x) \to R(x,x) \mid R \in \mathbf{R}_{\mathcal{O}}\} \\
\mathsf{Named}_{\mathcal{O}} = {} & \{\mathsf{Named}(a), \top(a) \mid a \in \mathbf{I}_{\mathcal{O}}\} \\
\mathsf{Bottom}_{\mathcal{O}} = {} & \{\bot \to A(t) \mid A \in \mathbf{C}_{\mathcal{O}}\}
\end{aligned}
$$

Listing 3 Auxiliary Rules. In the above, $\mathbf{C}_{\mathcal{O}}$, $\mathbf{R}_{\mathcal{O}}$, and $\mathbf{I}_{\mathcal{O}}$ are sets of all concept names, roles and individuals occurring in $\mathcal{O}$.

tively. The predicate Named, employed in the set $\mathsf{Named}_{\mathcal{O}}$, will be used to label the terms corresponding to individuals from the ontology $\mathcal{O}$ (as opposed to the constants f_{RC}) which, in turn, is necessary to correctly axiomatize the meaning of nominal schemas. Note that the rules $\mathsf{Self}_{\mathcal{O}}$ do not treat anonymous reflexive pairs (f_{RC}, f_{RC}) the same as named ones. This is consistent with the intuitive meaning of the translation: if t is not a singleton, $R^M(t,t)$ does not imply $t \subseteq (\exists R.\mathsf{Self})^M$.

Definition 5 Consider an $\mathcal{EL}^{++}$ ontology $\mathcal{O}$ in normal form and the function δ from Listing 1. Then, let $\mathcal{Q}_{\mathcal{O}} = \bigcup_{\alpha \in \mathcal{O}} \delta(\alpha) \cup \mathsf{Aux}_{\mathcal{O}}$.

Theorem 1 *Consider some $\mathcal{EL}^{++}$ ontology $\mathcal{O}$ in normal form. Then, $\mathcal{O}$ is unsatisfiable if and only if $\bot \in \mathsf{Ch}(\mathcal{Q}_{\mathcal{O}})$. Provided $\mathcal{O}$ is satisfiable, $\mathcal{O} \models \alpha$ if and only if $\delta(\alpha) \in \mathsf{Ch}(\mathcal{Q}_{\mathcal{O}})$ for any non-negated assertion α.*

Proof We will prove the theorem in two parts. First, consider a normalized $\mathcal{EL}^{++}$ ontology $\mathcal{O}$. If $\bot \in \mathsf{Ch}(\mathcal{Q}_{\mathcal{O}})$, $\mathcal{O}$ is unsatisfiable. Moreover, for any non-negated assertion α such that $\delta(\alpha) \in \mathsf{Ch}(\mathcal{Q}_{\mathcal{O}})$, $\mathcal{O} \models \alpha$.

Let the function κ map a constant t to a concept expression, as follows: if t is of the form f_{RD}, $\kappa(t) = \mathsf{Ran}(R) \sqcap D$; if $t = a \in \mathbf{I}_{\mathcal{O}}$, $\kappa(t) = \{a\}$. In a given model of $\mathcal{O}$, this concept expression becomes the set of model elements that t "represents", according to the intuitive meaning of the rule translation.

We make use of an inductive argument on the chase sequence $\mathcal{Q}^0_{\mathcal{O}}$, $\mathcal{Q}^1_{\mathcal{O}}, \ldots$ of $\mathcal{Q}_{\mathcal{O}}$ to establish the following claims.

1. If $\bot \in \mathcal{Q}^i_{\mathcal{O}}$, we have that $\mathcal{O}$ is unsatisfiable.
2. For all $A(t) \in \mathcal{Q}^i_{\mathcal{O}}$, we have that $\mathcal{O} \models \kappa(t) \sqsubseteq A$.
3. For all $R(t,u) \in \mathcal{Q}^i_{\mathcal{O}}$, we have that $\mathcal{O} \models \kappa(t) \sqsubseteq \exists R.\kappa(u)$.
4. For all $R(t,u) \in \mathcal{Q}^i_{\mathcal{O}}$ and $\mathsf{Ran}(S) \sqsubseteq A \in \mathcal{O}$ with $R \sqsubseteq^*_{\mathcal{O}} S$, we have that $\mathcal{O} \models \kappa(u) \sqsubseteq A$.
5. For all $R_{\mathsf{Self}}(t) \in \mathcal{Q}^i_{\mathcal{O}}$, we have that $\mathcal{O} \models \kappa(t) \sqsubseteq \exists R.\mathsf{Self}$.
6. For all $t \approx u \in \mathcal{Q}^i_{\mathcal{O}}$, we have that $\mathcal{O} \models \kappa(t) \equiv \kappa(u)$.
7. For all $t \in \mathbf{T}$ occurring in $\mathcal{Q}^i_{\mathcal{O}}$, we have that $\kappa(t)^{\mathcal{I}} \neq \emptyset$ for all $\mathcal{I} \models \mathcal{O}$.

The result follows from claims (1-3); the additional statements are used to properly establish the inductive step. The base case of the induction is straightforward as we have that $\mathcal{Q}^0_{\mathcal{O}} = \emptyset$. Most of the cases to be considered for the proof of the induction step are trivial, and therefore we do not include them in the following case by case analysis.

We verify that (4) holds in the induction step. Let $R(t,u) \in \mathcal{Q}^i_{\mathcal{O}}$, $(\mathsf{Ran}(S) \sqsubseteq A) \in \mathcal{O}$ and $R \sqsubseteq^*_{\mathcal{O}} S$ for some $R, S \in \mathbf{R}$ and some $t, u \in \mathbf{T}$. Since $R(t,u) \in \mathcal{Q}^i_{\mathcal{O}}$, then one of the following cases must hold.

- $R(t,u) \in \mathcal{Q}^{i-1}_{\mathcal{O}}$. By induction hypothesis (IH) (4), $\mathcal{O} \models \kappa(u) \sqsubseteq A$.
- $(\rightarrow R(t,u)) \in \mathcal{Q}_{\mathcal{O}}$. So t, u are not of the form f_{RD} and $R(t,u) \in \mathcal{O}$. Thus $\kappa(u) = \{u\}, \mathcal{O} \models \{u\} \sqsubseteq \mathsf{Ran}(R), R \sqsubseteq S$, and thus $\mathcal{O} \models \kappa(u) \sqsubseteq \mathsf{Ran}(S) \sqsubseteq A$.
- The constant u is of the form f_{RB}, there is a rule of the form $C(x) \rightarrow R(x, f_{RB}) \wedge B(f_{RB}) \in \mathcal{Q}_{\mathcal{O}}$ and $C(t) \in \mathcal{Q}^{i-1}_{\mathcal{O}}$. By definition, $\kappa(u) = B \sqcap \mathsf{Ran}(R)$. Since $R \sqsubseteq^*_{\mathcal{O}} S$, $\mathcal{O} \models \kappa(u) \sqsubseteq \mathsf{Ran}(S)$. Therefore, $\mathcal{O} \models \kappa(u) \sqsubseteq A$.
- $V(x,y) \rightarrow R(x,y) \in \mathcal{Q}_{\mathcal{O}}$ for some $V \in \mathbf{R}$, and so $V \sqsubseteq R \in \mathcal{O}$. Thus $V \sqsubseteq^*_{\mathcal{O}} S$. Now $V(t,u) \in \mathcal{Q}^{i-1}_{\mathcal{O}}$, so by IH (4), $\mathcal{O} \models \kappa(u) \sqsubseteq A$.
- There is a rule of the form $R_1(x_0,x_1) \wedge \ldots \wedge R_k(x_{k-1},x_n) \rightarrow R(x_0,x_k) \in \mathcal{Q}_{\mathcal{O}}$ with $k > 1$ and there are some terms $v_0, \ldots, v_k$ such that $\{R_1(v_0,v_1), \ldots, R_k(v_{k-1},v_k)\} \subseteq \mathcal{Q}^{i-1}_{\mathcal{O}}$, $v_0 = t$ and $v_k = u$. Then $\mathsf{Ran}(R_k) \sqsubseteq A \in \mathcal{O}$ by the restrictions defined in Section 2.1 for $\mathcal{ELV}^{++}$ ontologies. By IH (4), $\mathcal{O} \models \kappa(u) \sqsubseteq A$.
- $R_{\mathsf{Self}}(x) \rightarrow R(x,x) \in \mathcal{Q}_{\mathcal{O}}$, $R_{\mathsf{Self}}(t) \in \mathcal{Q}^{i-1}_{\mathcal{O}}$ and $t = u$. By IH (5), $\mathcal{O} \models \kappa(t) \sqsubseteq \exists R.\mathsf{Self}$. Therefore, $\mathcal{O} \models \kappa(u) \sqsubseteq \mathsf{Ran}(R)$. Since $R \sqsubseteq^*_{\mathcal{O}} S$, $\mathcal{O} \models \kappa(u) \sqsubseteq \mathsf{Ran}(S)$. Therefore, $\mathcal{O} \models \kappa(u) \sqsubseteq A$.
- $\{R(t,v), v \approx u\} \subseteq \mathcal{Q}^{i-1}_{\mathcal{O}}$ for some v. By IH (4) and (6), $\mathcal{O} \models \{\kappa(v) \sqsubseteq A, \kappa(v) \equiv \kappa(u)\}$. Hence, $\mathcal{O} \models \kappa(u) \sqsubseteq A$.
- $\{R(v,u), v \approx t\} \subseteq \mathcal{Q}^{i-1}_{\mathcal{O}}$ for some v. By IH (4), $\mathcal{O} \models \kappa(u) \sqsubseteq A$.

We will also show that IH (7) holds in the inductive step. If $t \in \mathbf{I}$, then the claim holds since $\kappa(t) = \{t\}$. Hence, we only need to consider the cases where $t \notin \mathbf{I}$.

- $t \in \mathcal{Q}^{i-1}_{\mathcal{O}}$. Then $\kappa(t)^{\mathcal{I}} \neq \emptyset$ by IH (7).
- The term t is of the form f_{RD}, there is a rule of the form $A(x) \rightarrow R(x, f_{RB}) \wedge B(f_{RB}) \in \mathcal{Q}_{\mathcal{O}}$, and $A(u) \in \mathcal{Q}^{i-1}_{\mathcal{O}}$ for some $u \in \mathbf{T}$. Then, $A \sqsubseteq \exists R.B \in \mathcal{O}$ and $\kappa(t) = \mathsf{ran}(R) \sqcap B$. By IH (2), $\mathcal{O} \models \kappa(u) \sqsubseteq A$. By IH (7), $\kappa(u)^{\mathcal{I}} \neq \emptyset$ for all $\mathcal{I} \models \mathcal{O}$. Therefore, $A^{\mathcal{I}}$ is non-empty as well for all $\mathcal{I} \models \mathcal{O}$, so $\exists R.B^{\mathcal{I}}$ is non-empty, and therefore so is $(\mathsf{Ran}(R) \sqcap D)^{\mathcal{I}} = \kappa(t)^{\mathcal{I}}$.

Now to the second part: consider a normalized $\mathcal{EL}^{++}$ ontology $\mathcal{O}$. If $\mathcal{O}$ is unsatisfiable, $\bot \in \mathsf{Ch}(\mathcal{Q}_{\mathcal{O}})$. Moreover, if $\mathcal{O}$ is satisfiable and $\mathcal{O} \models \alpha$ for any non-negative assertion α, then $\delta(\alpha) \in \mathsf{Ch}(\mathcal{Q}_{\mathcal{O}})$.

If $\bot \in \mathsf{Ch}(\mathcal{Q}_{\mathcal{O}})$, then the above claim trivially holds since $\mathcal{O}$ is unsatisfiable by the first part of the proof, above. Therefore, we assume that $\bot \notin \mathsf{Ch}(\mathcal{Q}_{\mathcal{O}})$. To show that the claim holds in this case, we construct a model $\mathcal{M} = (\Delta^{\mathcal{M}}, \cdot^{\mathcal{M}})$ for $\mathcal{O}$ such that,

(a) if $A(a) \notin \mathsf{Ch}(\mathcal{Q}_{\mathcal{O}})$, then $a^{\mathcal{M}} \notin A^{\mathcal{M}}$ (and hence $\mathcal{O} \not\models A(a)$); and

(b) if $R(a,b) \notin \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$, then $(a^\mathcal{M}, b^\mathcal{M}) \notin R^\mathcal{M}$ (and hence $\mathcal{O} \not\models R(a,b)$).

The model $\mathcal{M} = (\Delta^\mathcal{M}, \cdot^\mathcal{M})$ is defined as follows:

- Let Δ^* be the set containing every individual in $\mathbf{I}_\mathcal{O}$ and every term of the form f_{RB} which occurs in $\mathsf{Ch}(\mathcal{Q}_\mathcal{O})$.
- Let Δ^{**} be the set of equivalence classes of the relation $\equiv$ on $\Delta^* \times \Delta^*$ which is defined by $\{(a,b) \mid a \approx b \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})\}$. That this is an equivalence relation follows from the rules in $\mathsf{Eq}_\mathcal{O}$. Also note that we can safely abuse the notation "$A(t) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$" for an element $t \in \Delta^{**}$ to mean "$A(t') \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$ for some $t' \in t$", because the substitution rules in $\mathsf{Eq}_\mathcal{O}$ guarantee that any one element t' of the equivalence class t has $A(t') \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$ iff all elements do, and likewise for binary predicates; that is to say, $\equiv$ is a congruence in $\mathsf{Ch}(\mathcal{Q}_\mathcal{O})$. Recall our earlier decision with regard to reflexive pairs (x,x) - we use a special class R_{Self} for each role R to track where they are. But in the ruleset $\mathsf{Self}_\mathcal{O}$, we cautiously chose never to assume any reflexives exist except between named individuals, since other constants represent sets, potentially full of indistinguishable members, and even when such a set is "related to itself by R", it may be that none of its members are individually so related to themselves. So now we are faced with a chase set constructed by this very conservative assumption; we need a model $\mathcal{M}$ that justifies it. Therefore we will ensure that each non-named constant really corresponds to a subset of $\Delta^\mathcal{M}$ with at least 2 elements, with no reflexive connection between them by any *simple* role. This is done by "exploding" each non-named constant t into two, t and t^*, such that if $S(t,t)$ holds in the chase, S will connect t to t^* and vice versa, but not t to t or t^* to t^*. Now the semantics of $\mathcal{EL}^{++}$ are still capable of forcing some roles in $\mathcal{M}$ to contain reflexive pairs - if, e.g., $S \circ S \sqsubseteq R$ were an axiom of $\mathcal{O}$. But such R would not be simple, and $\exists R.\mathsf{Self}$ would not be defined. Simple roles thus serve to prevent an undesirable interaction between role chains and the Self construct. More precisely: let $X \subseteq \Delta^{**}$ be the set of singleton $\equiv$-classes of the form $\{f_{RB}\}$. We just state the following fact: for all $t \in \Delta^{**}$, either $t \in X$ or (exclusive "or") t contains some individual ($\in \mathbf{I}$). To see that this is true, notice the ways in which $\approx$ atoms involving non-individuals can be introduced by our rules. Now the model will be constructed using the set $\Delta^\mathcal{M} = (\Delta^{**} \setminus X) \cup (X \times \{0,1\})$. For $t \in \Delta^\mathcal{M}$, let its projection ιt be defined by $[f_{RB}]_\equiv$ if $t = (\{f_{RB}\}, i) \in X$, and by t otherwise. For $t = (\{f_{RB}\}, i) \in X$, let $t^* = (\{f_{RB}\}, 1-i)$.
- The function $\cdot^\mathcal{M}$ is defined as follows:
  1. For all $a \in \mathbf{I}$, $a^\mathcal{M} = [a]_\equiv$.
  2. For all $A \in \mathbf{C}$, $A^\mathcal{M} = \{t \mid A(\iota t) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})\}$.
  3. For all simple $S \in \mathbf{R}$, $S^\mathcal{M} = \{(t,t) \mid S_{\mathsf{Self}}(\iota t) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})\} \cup \{(t,u) \mid S(\iota t, \iota u) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O}) \text{ and } t \neq u\}$.
  4. For all non-simple $R \in \mathbf{R}$, $R^\mathcal{M} = \{(t,u) \mid R(\iota t, \iota u) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})\}$.

By (1), $\top^\mathcal{M} = \Delta^\mathcal{M}$ since $\top(c) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$ for every constant c occurring in $\mathsf{Ch}(\mathcal{Q}_\mathcal{O})$ (note that $\mathsf{Top}_\mathcal{O} \subseteq \mathcal{Q}_\mathcal{O}$). Since $\bot$ is a null-ary symbol in our Datalog

signature, no $\bot(t) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$, so $\bot^\mathcal{M} = \emptyset$. Therefore, $\mathcal{M}$ is indeed an interpretation. Moreover, condition (a) above is satisfied, since if $A(a) \notin \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$, $[a]_\equiv \notin A^\mathcal{M}$. For condition (b), let $R(t,u) \notin \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$ for *individuals* t, u; if R is non-simple then $([t]_\equiv, [u]_\equiv) \notin R^\mathcal{M}$ immediately; whereas if R is simple, this is immediate if $[t]_\equiv \neq [u]_\equiv$, that is, $t \approx u \notin \mathsf{Ch}(\mathcal{Q}_\mathcal{O}), t \neq u$. Otherwise, $(t^\mathcal{M}, u^\mathcal{M}) \in R^\mathcal{M}$ means $R_{\mathsf{Self}}(t) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$, from which would follow $R(t,t) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$ by a rule from $\mathsf{Self}_\mathcal{O}$, which implies $R^\mathcal{M}(t^\mathcal{M}, t^\mathcal{M})$, a contradiction. So condition (b) holds as well.

To show that $\mathcal{M}$ is a model, one verifies that $\mathcal{M}$ satisfies every axiom $\alpha \in \mathcal{O}$. We show this for the axiom shapes corresponding to lines 3, 6, 10, 11, and 15 in Listing 1.

- Let $\exists S.\mathsf{Self} \sqsubseteq B \in \mathcal{O}$ and $(t,t) \in S^\mathcal{M}$. Then, S is simple by the definition of an $\mathcal{EL}^{++}$ ontology and hence, $S_{\mathsf{Self}}(\iota t) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$. Since $S_{\mathsf{Self}}(x) \to B(x) \in \mathcal{Q}_\mathcal{O}$, $B(\iota t) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$ and hence, $t \in B^\mathcal{M}$.
- Let $A \sqsubseteq \exists R.B \in \mathcal{O}$ and $t \in A^\mathcal{M}$. Then, $A(\iota t) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$. Since $A(x) \to R(x, f_{RB}) \wedge B(f_{RB}) \in \mathcal{Q}_\mathcal{O}$, $R(\iota t, [f_{RB}]_\equiv), B([f_{RB}]_\equiv) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$. We consider the following cases:
  - $t = [f_{SB}]_\equiv$ and this set is not a singleton: Then $[f_{SB}]_\equiv$ contains an individual a, and $\iota t = t$. $\mathsf{Named}(t), \mathsf{Self}_R(t) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$, so regardless of whether R is simple, $(t,t) \in R^\mathcal{M}$.
  - $t \in X$: Then regardless of whether R is simple, $(t, t^*), (t^*, t) \in R^M$, and $t, t^* \in B^M$.
  - $t \neq [f_{RB}]_\equiv$: Then, $(t, f_{SB}) \in S^\mathcal{M}$ by definition.
- Let $S \sqsubseteq R \in \mathcal{O}$ and $(t,u) \in S^\mathcal{M}$. Then $\{S_{\mathsf{Self}}(x) \to R_{\mathsf{Self}}(x), S(x,y) \to R(x,y)\} \subseteq \mathcal{Q}_\mathcal{O}$. The following cases arise:
  - S is non-simple and $S(\iota t, \iota u) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$. Then, R is also non-simple. Also, $R(\iota t, \iota u) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$ and hence, $(t,u) \in R^\mathcal{M}$.
  - S is simple and $t \neq u$, and $S(\iota t, \iota u) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$. Then, $R(\iota t, \iota u) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$ and $(t,u) \in R^\mathcal{M}$.
  - S is simple, $S_{\mathsf{Self}}(t) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$ and $t = u$. Then, $R_{\mathsf{Self}}(\iota t), R(\iota t, \iota t) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$ and so regardless of whether R is simple $(t,t) \in R^\mathcal{M}$.
- Let $R_1 \circ \ldots \circ R_m \sqsubseteq S \in \mathcal{O}$ with $m > 1$, $(v_0, v_1) \in R_1^\mathcal{M}, \ldots$, and $(v_{m-1}, v_m) \in R_m^\mathcal{M}$. Then, $R_1(x_0, x_1) \wedge \ldots \wedge R_m(x_{m-1}, x_m) \to S(x_0, x_m) \in \mathcal{Q}_\mathcal{O}$, S is complex, and $R(\iota v_0, \iota v_1), \ldots, R(\iota v_{m-1}, \iota v_m) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$. Hence, $S(\iota v_0, \iota v_m) \in \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$ and $(v_0, v_m) \in S^\mathcal{M}$.
- Let $\neg R(a,b) \in \mathcal{O}$. Then $R(a,b) \to \bot \in \mathcal{Q}_\mathcal{O}$. Moreover, by assumption, $\bot \notin \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$ and hence $R(a,b) \notin \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$. We consider the following cases:
  - R is complex or $a \neq b$. Then, $R(a,b) \notin \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$ implies $(a^\mathcal{M}, b^\mathcal{M}) \notin R^\mathcal{M}$.
  - R is simple and $a = b$. Note that $R_{\mathsf{Self}}(a) \notin \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$ as this would imply that $R(a,b) \notin \mathsf{Ch}(\mathcal{Q}_\mathcal{O})$ since $R_{\mathsf{Self}}(x) \to R(x,x) \in \mathcal{Q}_\mathcal{O}$. Hence $(a^\mathcal{M}, b^\mathcal{M}) \notin R^\mathcal{M}$.

□

It follows from Theorem 1 that we can use the chase over the Datalog program $\mathcal{Q}_\mathcal{O}$ to solve reasoning tasks over a normalized ontology $\mathcal{O}$ in the

obvious way. In particular, we can also decide whether $\mathcal{O}$ entails a negated assertion $\neg\alpha$, by constructing the chase $\mathsf{Ch}(\mathcal{Q}_{\mathcal{O}'})$ where $\mathcal{O}' = \mathcal{O} \cup \{\alpha\}$. $\mathcal{O} \models \neg\alpha$ iff this chase contains $\bot$.

Most of the mappings performed by δ, depicted in Listing 1, are well-known translations from DL axioms into first-order predicate logic (FOL) formulas (with universal quantifiers implicit). The main exceptions are the axioms of the form $A \sqsubseteq \exists R.B$ and axioms featuring occurrences of the Self constructor. Instead of introducing fresh constants or using function symbols with non-zero arity, we reuse the same constant f_{RB} to "satisfy" all existential restrictions enforced by axioms of the form $A \sqsubseteq \exists R.B$.

4 A Datalog Algorithm for $\mathcal{ELV}^{++}$ Ontologies

4.1 Nominal Schemas and Grounding

So far, we have provided an algorithm to reason over normalized $\mathcal{EL}^{++}$ ontologies. Theoretically, this procedure can be used to reason over $\mathcal{ELV}^{++}$ ontologies since nominal schemas can be normalized away using nominals [29]. That is, an axiom in an ontology featuring nominal schemas such as

$$\begin{aligned}&\exists \text{hasReviewAssignment}.((\{x\} \sqcap \exists \text{hasAuthor}.\{y\}) \sqcap (\{x\} \sqcap \exists \text{atVenue}.\{z\}))\\ &\quad \sqcap \exists \text{hasSubmittedPaper}.(\exists \text{hasAuthor}.\{y\} \sqcap \exists \text{atVenue}.\{z\})\\ &\quad \sqsubseteq \exists \text{hasConflictingAssignedPaper}.\{x\}\end{aligned}$$

can be substituted by all of the axioms in the set

$$\begin{aligned}\{&\exists \text{hasReviewAssignment}.((\{a\} \sqcap \exists \text{hasAuthor}.\{b\}) \sqcap (\{a\} \sqcap \exists \text{atVenue}.\{c\}))\\ &\quad \sqcap \exists \text{hasSubmittedPaper}.(\exists \text{hasAuthor}.\{b\} \sqcap \exists \text{atVenue}.\{c\})\\ &\quad \sqsubseteq \exists \text{hasConflictingAssignedPaper}.\{a\} \mid a, b, c \in \mathbf{I}_{\mathcal{O}}\}\end{aligned}$$

yielding an ontology which entails the same assertions, so it can be used in place of the original ontology for both our reasoning tasks, satisfiability and assertion retrieval.

Definition 6 Recall that $\mathbf{I}_{\mathcal{O}}$ denotes the set of all individuals occurring in an ontology $\mathcal{O}$. Given an axiom α, $\alpha_{\mathtt{g}}$ is the set of all axioms that can be constructed by uniformly replacing all occurrences of each nominal schema in α with some individual from $\mathbf{I}_{\mathcal{O}}$. Given an ontology $\mathcal{O}$, let $\mathcal{O}_{\mathtt{g}} = \bigcup_{\alpha \in \mathcal{O}} \alpha_{\mathtt{g}}$.

Lemma 1 *An ontology $\mathcal{O}$ entails an axiom α if and only if $\mathcal{O}_g \models \alpha$.*

Proof We prove the equivalent claim that $\mathcal{O}$ and $\mathcal{O}_{\mathtt{g}}$ have the same models $\mathcal{I}$. We take it as evident that if $\alpha_{\mathcal{Z}}$ is the axiom formed from α by replacing each nominal variable v with $\mathcal{Z}(v)$, $\mathcal{I}, \mathcal{Z} \models \alpha$ iff $\mathcal{I}, \mathcal{Z}^* \models \alpha'$ for *all* nominal variable assignments $\mathcal{Z}^*$ (the satisfaction of α' not depending on any assignment of nominal variables). Let $\mathcal{I} \models \mathcal{O}$, so $\mathcal{I}, \mathcal{Z} \models \mathcal{O}$ for all $\mathcal{Z}$. Let $\alpha' \in \mathcal{O}_{\mathtt{g}}$. Then α'

is clearly $\alpha_{\mathcal{Z}}$ for some $\mathcal{Z}$ and $\alpha \in \mathcal{O}$ and thus since $\mathcal{I}, \mathcal{Z} \models \alpha$, $\mathcal{I}, \mathcal{Z}^* \models \alpha'$ for all $\mathcal{Z}^*$, so $I \models \alpha'$. Conversely: Let $\alpha \in \mathcal{O}$, $I \models \mathcal{O}_{\mathsf{g}}$, and $\mathcal{Z}$ be any assignment. Then for all $\mathcal{Z}^*$, $\mathcal{I}, \mathcal{Z}^* \models \alpha_{\mathcal{Z}}$, since $\alpha_{\mathcal{Z}} \in \mathcal{O}_{\mathsf{g}}$. Therefore $\mathcal{I}, \mathcal{Z} \models \alpha$. □

Unsurprisingly, grounding all axioms containing nominal schemas does not result in a very efficient procedure [8]. Note that, for some axiom α in an ontology $\mathcal{O}$, $|\alpha_{\mathsf{g}}| = |\mathbf{I}_{\mathcal{O}}|^n$ with n the number of different nominal schemas in α. Even if the growth in the number of axioms resulting from the grounding of an $\mathcal{ELV}^{++}$ is polynomial (since there is an upper bound n on the number of different nominal schemas per axiom) the number of new axioms in $\mathcal{O}_{\mathsf{g}}$ increases substantially for real-world, data-intensive ontologies that may contain millions of individuals. To circumvent this issue, in the following section we introduce an algorithm that does not make use of grounding.

4.2 The Translation

We present a function that maps (possibly not normalized) $\mathcal{ELV}^{++}$ axioms to Datalog rules. As in Section 3, we make use of this function to map $\mathcal{ELV}^{++}$ ontologies to Datalog programs that can be used to solve satisfiability and assertion retrieval. Foregoing the need for an intermediate normalization step was necessary for us since we could not find an efficient way to normalize $\mathcal{ELV}^{++}$ axioms similar to what is usually done with $\mathcal{EL}^{++}$. Recall the definition of $\mathsf{f}(\exists R.C)$ from section 2.2, which defines a term from an existential concept expression. The resulting term, $f_{RC_*}(\vec{t})$, where $\vec{t}$ is the vector of individuals and nominal variables in C, has been ground up to now; now we consider $\mathcal{ELV}^{++}$ and so class expressions for which this term will not be ground (recall nominal variables in a DL signature are re-used as *variables* in our Datalog signature.)

Definition 7 Given some $\mathcal{ELV}^{++}$ ontology $\mathcal{O}$, let $\mathcal{P}_{\mathcal{O}} = \bigcup_{\alpha \in \mathcal{O}} \gamma(\alpha) \cup \mathsf{Aux}_{\mathcal{O}}$ with γ the function from Listing 4 and $\mathsf{Aux}_{\mathcal{O}}$ the rule set defined in Listing 3.

Example 4 Let α be the DL axiom containing nominal schemas presented in Section 4.1. Then, $\gamma(\alpha)$ is the singleton set containing the following rule.

$$\begin{aligned}\text{hasReviewAssignment}(w, w_1) \wedge w_1 \approx x \wedge \mathsf{Named}(x) \wedge \\ \text{hasAuthor}(w_1, w_2) \wedge w_2 \approx y \wedge \mathsf{Named}(y) \wedge \\ \text{atVenue}(w_1, w_3) \wedge w_3 \approx z \wedge \mathsf{Named}(z) \wedge \\ \text{hasSubmittedPaper}(w, w_4) \wedge \text{hasAuthor}(w_4, w_5) \wedge w_5 \approx y \wedge \\ \text{atVenue}(w_4, w_6) \wedge w_6 \approx z \rightarrow \text{hC}(w, f_{\exists \text{hC}.\{\star\}}(x)) \wedge f_{\exists \text{hC}.\{\star\}}(x) \approx x\end{aligned}$$

In the above, hC is a shortcut for the role hasConflictingAssignedPaper

We proceed with an intermediate result that illustrates how the program $\mathcal{P}_{\mathcal{O}}$ may be used to solve reasoning tasks over $\mathcal{EL}^{++}$ ontologies.

$$
\begin{aligned}
C \equiv D &\mapsto \gamma(C \sqsubseteq D) \cup \gamma(D \sqsubseteq C)\\
C \sqsubseteq D &\mapsto \{\gamma_l(C, w) \cup \mathsf{Named}(C) \cup \mathsf{Named}(D) \rightarrow \gamma_r(D, w)\}\\
\mathsf{Dom}(R) \sqsubseteq D &\mapsto \gamma(\exists R.\top \sqsubseteq D)\\
\mathsf{Ran}(R) \sqsubseteq D &\mapsto \{\{R(x, w)\} \cup \mathsf{Named}(D) \rightarrow \gamma_r(D, w)\}\\
\mathsf{Ref}(R) &\mapsto \{\top(x) \rightarrow R_{\mathsf{Self}}(x)\}\\
R \equiv S &\mapsto \gamma(R \sqsubseteq S) \cup \gamma(S \sqsubseteq R)\\
R \sqsubseteq S &\mapsto \{R(x, y) \rightarrow S(x, y), R_{\mathsf{Self}}(x) \rightarrow S_{\mathsf{Self}}(x)\}\\
R_1 \circ \ldots \circ R_n \sqsubseteq S &\mapsto \{R_1(x_0, x_1) \wedge \ldots \wedge R_n(x_{n-1}, x_n) \rightarrow S(x_0, x_n)\}\\
\mathsf{Tran}(R) &\mapsto \{R(x, y) \wedge R(y, z) \rightarrow R(x, z)\}\\
C(a) &\mapsto \{\mathsf{Named}(C) \rightarrow \gamma_r(C, a)\}\\
\neg C(a) &\mapsto \{\mathsf{Named}(C) \cup \gamma_l(C, a) \rightarrow \bot\}\\
R(a, b) &\mapsto \{R(a, b)\}\\
\neg R(a, b) &\mapsto \{R(a, b) \rightarrow \bot\}\\
a \approx b &\mapsto \{a \approx b\}\\
a \not\approx b &\mapsto \{a \approx b \rightarrow \bot\}
\end{aligned}
$$

$$
\begin{aligned}
\gamma_*(C_1 \sqcap \ldots \sqcap C_n, t) &\mapsto \gamma_*(C_1, t) \cup \ldots \cup \gamma_*(C_n, t)\\
\gamma_r(\exists R.C, t) &\mapsto R(t, \mathsf{f}(\exists R.C)) \cup \gamma_r(C, \mathsf{f}(\exists R.C))\\
\gamma_l(\exists R.C, t) &\mapsto \{R(t, w)\} \cup \gamma_l(C, w)\\
\gamma_*(\exists R.\mathsf{Self}, t) &\mapsto \{R_{\mathsf{Self}}(t)\}
\end{aligned}
\qquad
\begin{aligned}
\gamma_*(A, t) &\mapsto \{A(t)\}\\
\gamma_*(\bot, t) &\mapsto \{\bot\}\\
\gamma_*(\{o\}, t) &\mapsto \{o \approx t\}
\end{aligned}
$$

Listing 4 Functions γ, γ_l, and γ_r. In the above, $A \in \mathbf{C} \setminus \{\bot\}$; $C_{(i)}, D \in \mathbf{E}$; $R_{(i)}, S \in \mathbf{R}$; $a, b \in \mathbf{I}$; $o \in \mathbf{I} \cup \mathbf{N}$; $\mathsf{f}(\cdot)$ is the function from Definition 2; $\mathsf{Named}(\alpha) = \{\mathsf{Named}(v) \mid v \in \mathbf{N}$ occurring in $\alpha\}$ for any concept expression α; $x_{(i)}$, y, z and w are globally fresh variables; $n > 1$; and $* \in \{r, l\}$. Recall that sets of atoms indicate conjunctions.

Lemma 2 *An $\mathcal{EL}^{++}$ ontology $\mathcal{O}$ is unsatisfiable if and only if $\bot \in \mathsf{Ch}(\mathcal{P}_\mathcal{O})$. If $\mathcal{O}$ is satisfiable, then for any assertion α, $\mathcal{O} \models \alpha$ if and only if $\alpha \in \mathsf{Ch}(\mathcal{P}_\mathcal{O})$.*

Proof Let NF be the function from Listing 2. Given an $\mathcal{EL}^{++}$ ontology $\mathcal{O}$, let $\mathcal{O}_1, \ldots, \mathcal{O}_n$ be a sequence of ontologies such that $\mathcal{O}_n$ is a normalized ontology, $\mathcal{O}_1 = \mathcal{O}$ and, for all $i = 1, \ldots, n-1$, $\mathcal{O}_{i+1}$ is the ontology that results from substituting some axiom $\alpha \in \mathcal{O}_i$ not in normal form by the axioms in $\mathsf{NF}(\alpha)$. Moreover, let $\mathsf{A}(\mathcal{O}_i)$ be the smallest set of facts containing $\bot$ and every assertion that can be defined using the predicates and individuals in $\mathcal{O}_i$. Lemma 2 follows from the following claims:

1. For all $\alpha \in \mathsf{A}(\mathcal{O}_n)$, $\mathcal{O}_n \models \alpha$ if and only if $\alpha \in \mathsf{Ch}(\mathcal{P}_{\mathcal{O}_n})$ or $\bot \in \mathsf{Ch}(\mathcal{P}_{\mathcal{O}_n})$.
2. For all $i = 2, \ldots, n$ and $\alpha \in \mathsf{A}(\mathcal{O}_{i-1})$, $\mathcal{O}_{i-1} \models \alpha$ if and only if $\mathcal{O}_i \models \alpha$.
3. For all $i = 2, \ldots, n$ and $\alpha \in \mathsf{A}(\mathcal{O}_{i-1})$, $\alpha \in \mathsf{Ch}(\mathcal{P}_{\mathcal{O}_{i-1}})$ iff $\alpha \in \mathsf{Ch}(\mathcal{P}_{\mathcal{O}_i})$.

Claim (1) holds by Theorem 1, which can be applied to an ontology $\mathcal{O}_n$ in normal form (note that, $\mathcal{Q}_{\mathcal{O}_n}$ is equal to $\mathcal{P}_{\mathcal{O}_n}$ up to renaming of the variables in the rules, and hence, $\mathsf{Ch}(\mathcal{Q}_{\mathcal{O}_n}) = \mathsf{Ch}(\mathcal{P}_{\mathcal{O}_n})$). Claim (2) follows directly from Proposition 1. Claim (3) can be verified by showing that, for any $i = 2, \ldots, n$,

the defined condition holds. Let $\mathcal{O}_i$ be the ontology that results from replacing an axiom $\alpha \in \mathcal{O}_{i-1}$ of the form (j) as defined in Listing 2 by the axioms in the set $\mathsf{NF}(\alpha)$. One can proceed with a case by case analysis to show that, irrespective of the type of axiom (j), claim (3) holds for $\mathcal{O}_{i-1}$ and $\mathcal{O}_i$.

- Let $j \in \{1, 2, 4, 5, 6, 10, 13\}$. Then, $\mathcal{P}_{\mathcal{O}_{i-1}} = \mathcal{P}_{\mathcal{O}_i}$.
- Let $j \in \{3, 7, 9, 11, 12, 15, 16\}$. In this case, a set of axioms including some fresh concept names $X_{(\ell)}$ are introduced in $\mathcal{O}_i$. $\mathsf{Ch}(\mathcal{P}_{\mathcal{O}_{i-1}})$ is identical to the restriction of $\mathsf{Ch}(\mathcal{P}_{\mathcal{O}_i})$ to those facts not using fresh predicates, up to replacement of some constants f_{RD} by fresh constants f_{RX}. Since the f_{RD} do not occur in assertions, the claim holds.
- Let $j = \{8, 14\}$. In this case, $\mathsf{Ch}(\mathcal{P}_{\mathcal{O}_{i-1}}) = \mathsf{Ch}(\mathcal{P}_{\mathcal{O}_i})$

□

Applying the previous result, we verify that the program $\mathcal{P}_{\mathcal{O}}$ can be used to solve reasoning tasks over the $\mathcal{ELV}^{++}$ ontology $\mathcal{O}$.

First it will be necessary to develop some more technical properties of the translation γ.

If C is a concept expression and σ a substitution defined on the nominal variables of C and mapping to individuals, let $C\sigma$ be the concept expression C with each nominal variable v replaced by $\sigma(v)$. The key property of the functional term $\mathsf{f}(\exists R.D)$ is that if $\mathsf{f}(\exists R.D) = f_{RD*}(\vec{t})$, then $\mathsf{f}(\exists (R.D)\sigma) = f_{RD*}(\vec{t\sigma})$. It will be invoked to prove Lemmas 4 and 5.

Lemma 3 *For any $\mathcal{ELV}^{++}$ concept expression C, term t over only nominal variables and ground substitution σ defined only on nominal variables and defined on all nominal variables in C and t, $\gamma_r(C, t)\sigma = \gamma_r(C\sigma, t\sigma)$, and contains no fresh variables.*

Proof Strategy By induction on the 6 cases in the definition of γ_r.

Lemma 4 *For any $\mathcal{ELV}^{++}$ concept expression C and ground substitution σ defined only on nominal variables and defined on all nominal variables in C, $\gamma_*(C, t)\sigma \cong \gamma_*(C\sigma, u)$ for any fresh (non-nominal) variables t, u, by an isomorphism that maps t to u.*

Proof By induction on the seven cases in the definition of γ_*. Most cases are easy.

(1): $C = \sqcap_i C_i$. $\gamma_*(\sqcap_i, t)\sigma = \bigwedge_i \gamma_*(C_i, t)\sigma$. By inductive hypothesis, for each i, $\gamma_*(C_i, t)\sigma \cong \gamma_*(C_i\sigma, u)$ by an isomorphism taking t to u. Furthermore, by the standard assumption about fresh variable introduction, the $\gamma_*(C_i, t)\sigma$ have no fresh variables in common, and neither do the $\gamma_*(C_i\sigma, t)$, and none of these contain any nominal variables because σ annihilates them. So, the separate isomorphisms can be combined to yield one $i : \bigwedge_i \gamma_*(C_i, t)\sigma \cong \bigwedge_i \gamma_*(C_i\sigma, u)$ as required.

(2): C is of the form $\exists R.C$, $* = r$. So, $\gamma_r(\exists R.C, t)\sigma = [R(t, f_{RC_\star}(\vec{v}) \wedge \gamma_r(C, f_{RC_\star}(\vec{v}))]\sigma = R(t, f_{RC_\star}(\vec{v\sigma})) \wedge \gamma_r(C, f_{RC_\star}(\vec{v}))\sigma$. Using lemma 3, this is

$= R(t, f_{RC_\star}(\vec{v}\sigma)) \wedge \gamma_r(C\sigma, f_{RC_\star}(\vec{v}\sigma)) \cong R(u, f_{RC_\star}(\vec{v}\sigma)) \wedge \gamma_r(C\sigma, f_{RC_\star}(\vec{v}\sigma))$ certainly, because by lemma 3 no part of these conjunctions contains a fresh variable other than the single occurrence of t or u; and this last is $= \gamma_r(\exists R.C\sigma, u)$.

(3): C is of the form $\exists R.C$, $* = l$. So, $\gamma_l(\exists R.C, t)\sigma = [R(t, w) \wedge \gamma_l(C, w)]\sigma$ and, since w is not a nominal variable, this $= R(t, w) \wedge \gamma_l(C, w)\sigma$. Meanwhile, $\gamma_l(\exists R.C\sigma, u) = R(u, w') \wedge \gamma_l(C\sigma, w')$. Now by induction $\gamma_l(C, w)\sigma \cong \gamma_l(C\sigma, w')$ by an isomorphism mapping w to w'; and by the standard assumption about fresh variable introduction, the former does not contain t nor does the latter contain u, so this isomorphism can be extended with the mapping $t \mapsto u$ to obtain the required isomorphism.

The remaining cases are not shown. □

Lemma 5 *For any $\mathcal{ELV}^{++}$ axiom α, and substitution σ defined on exactly the nominal variables in α, such that $\sigma(v)$ is an individual for all such variables v, let $\alpha\sigma$ denote α with all nominal variables x replaced by $x\sigma$. For a rule $B \to H$, let $r(B \to H)$ denote the rule $B' \to H$ where B' is the subset of B containing atoms that are not over the predicate Named. Then $(r(\gamma(\alpha)))\sigma \cong \gamma(\alpha\sigma)$.*

Proof This proof also requires many cases, one for each line in the definition of γ. Throughout this proof we use N to denote any conjunction of "Named" atoms. The most important case is

(2): $\alpha = (C \sqsubseteq D)$. $(r(\gamma(C \sqsubseteq D)))\sigma = (r(\gamma_l(C, t) \wedge N \to \gamma_r(D, t)))\sigma = \gamma_l(C, t)\sigma \to \gamma_r(D, t)\sigma \cong \gamma_l(C\sigma, u) \to \gamma_r(D\sigma, u) = \gamma((C \sqsubseteq D)\sigma)$, as required. The isomorphism comes from lemma 4. There is no N in $\gamma((C \sqsubseteq D)\sigma)$ because $(C \sqsubseteq D)\sigma$ contains no nominal variables. Most of the other cases require no induction at all and are immediate; cases 1,3,4,6,10,11 follow from Lemma 4 similarly to case 2.

Lemma 6 *For any $\mathcal{ELV}^{++}$ ontology $\mathcal{O}$, $\mathsf{Ch}(\mathcal{P}_{\mathcal{O}_g}) = \mathsf{Ch}(\mathcal{P}_{\mathcal{O}})$.*

Proof We show first that $\mathsf{Ch}(\mathcal{P}_{\mathcal{O}}) \subseteq \mathsf{Ch}(\mathcal{P}_{\mathcal{O}_g})$, via induction on the chase sequence of $\mathcal{P}_{\mathcal{O}}$; namely, we verify that $\mathcal{P}^i_{\mathcal{O}} \subseteq \mathsf{Ch}(\mathcal{P}_{\mathcal{O}_g})$ for all $i \geq 0$. The base case of the induction trivially holds, since $\mathcal{P}^0_{\mathcal{O}} = \emptyset$. We proceed with the proof of the induction step.

Let α be a fact in $\mathcal{P}^i_{\mathcal{O}}$. Then, there must be some rule $R = B \to H \in \mathcal{P}_{\mathcal{O}}$ and some substitution σ with $B\sigma \subseteq \mathcal{P}^{i-1}_{\mathcal{O}}$ and $\alpha \in H\sigma$. Note that, for every nominal schema x in R, $\sigma(x)$ is an individual since $\mathsf{Named}(x) \in B$ and, for every term t, $\mathsf{Named}(t) \in \mathsf{Ch}(\mathcal{P}_{\mathcal{O}})$ iff $t \in \mathbf{I}_{\mathcal{O}}$. Suppose the rule R is $\gamma(\beta)$ for some axiom $\beta \in \mathcal{O}$. Let σ', σ'' be the restrictions of σ to nominal variables and non-nominal variables respectively. Then $\beta\sigma'$ is an axiom of $\mathcal{O}_g$, and so $\gamma(\beta\sigma') \in \mathcal{P}_{\mathcal{O}_g}$, and by Lemma 5, $(r(\gamma(\beta)))\sigma' \cong \gamma(\beta\sigma')$. Call the isomorphism ι.

Now σ'' maps the body of $\gamma(\beta)\sigma'$ into ${\mathcal{P}_{\mathcal{O}}}^{i-1} \subseteq \mathsf{Ch}(\mathcal{P}_{\mathcal{O}_g})$, and so also the smaller body of $(r(\gamma(\beta)))\sigma'$, so $\sigma'' \circ \iota^{-1}$ maps the body of $\gamma(\beta\sigma')$ into $\mathsf{Ch}(\mathcal{P}_{\mathcal{O}_g})$, and so also the head, since $\mathsf{Ch}(\mathcal{P}_{\mathcal{O}_g})$ is closed under the application of $\gamma(\beta\sigma')$. Now $(\sigma'' \circ \iota^{-1}) \circ \iota$ maps the head of $(r(\gamma(\beta)))\sigma'$ into $\mathsf{Ch}(\mathcal{P}_{\mathcal{O}_g})$, and that is to say σ maps the head of $\gamma(\beta)$ into $\mathsf{Ch}(\mathcal{P}_{\mathcal{O}_g})$, thus $\alpha \in \mathsf{Ch}(\mathcal{P}_{\mathcal{O}_g})$.

There remains the case that R is not $\gamma(\beta)$ for some axiom β, but is in $\mathsf{Aux}_{\mathcal{O}}$. Then R is in $\mathcal{P}_{\mathcal{O}_\mathsf{g}}$ as well, so $\alpha \in \mathsf{Ch}(\mathcal{P}_{\mathcal{O}_\mathsf{g}})$.

Second, we prove that $\mathcal{P}^i_{\mathcal{O}_\mathsf{g}} \subseteq \mathsf{Ch}(\mathcal{P}_{\mathcal{O}})$ for all $i \geq 0$. The base case is again trivial. Inductive step:

Let α be some fact occurring in $\mathcal{P}^i_{\mathcal{O}_\mathsf{g}}$. Then, there must be some rule $R = B \to H \in \mathcal{P}_{\mathcal{O}_\mathsf{g}}$ and some substitution σ with $B\sigma \subseteq \mathcal{P}^{i-1}_{\mathcal{O}}$ and $\alpha \in H\sigma$. Suppose R is of the form $\gamma(\delta)$ for some axiom δ of $\mathcal{O}_\mathsf{g}$. Then there is an axiom $\beta \in \mathcal{O}$ and a substitution σ' defined exactly on the nominal variables of β such that $\beta\sigma' = \delta$, and σ''s range consists of individuals. There is by Lemma 5 an isomorphism $\iota : (r(\gamma(\beta)))\sigma' \cong \gamma(\beta\sigma')$. Since σ maps the body of $\gamma(\beta\sigma')$ into $\mathsf{Ch}(\mathcal{P}_{\mathcal{O}})$, $\sigma \circ \iota$ maps the body of $(r(\gamma(\beta)))\sigma'$ into $\mathsf{Ch}(\mathcal{P}_{\mathcal{O}})$ and in fact the body of $\gamma(\beta))\sigma'$ as well, since $\mathsf{Ch}(\mathcal{P}_{\mathcal{O}})$ contains the ground atoms $\mathsf{Named}(a)$ for all a. Then, since $\mathsf{Ch}(\mathcal{P}_{\mathcal{O}})$ is closed under the application of the rule $\gamma(\beta)$, $\sigma \circ \iota$ maps the head of $\gamma(\beta)\sigma'$ into $\mathsf{Ch}(\mathcal{P}_{\mathcal{O}})$ and so $(\sigma \circ \iota) \circ \iota^{-1} = \sigma$, the head of $\gamma(\beta\sigma')$; and thus $\alpha \in \mathsf{Ch}(\mathcal{P}_{\mathcal{O}})$. Again if R is instead in $\mathsf{Aux}_{\mathcal{O}}$ the proof is trivial. □

Theorem 2 *An $\mathcal{ELV}^{++}$ ontology $\mathcal{O}$ is unsatisfiable if and only if $\bot \in \mathsf{Ch}(\mathcal{P}_{\mathcal{O}})$. If $\mathcal{O}$ is satisfiable, $\mathcal{O} \models \alpha$ if and only if $\alpha \in \mathsf{Ch}(\mathcal{P}_{\mathcal{O}})$ for every assertion α.*

Proof If $\mathcal{O}$ is satisfiable, $\mathcal{O} \models \alpha$ iff $\mathcal{O}_\mathsf{g} \models \alpha$ iff $\alpha \in \mathsf{Ch}(\mathcal{P}_{\mathcal{O}_\mathsf{g}})$ iff $\alpha \in \mathsf{Ch}(\mathcal{P}_{\mathcal{O}})$, by Lemmas 1, 2, and 6, respectively. $\mathcal{O}$ is unsatisfiable iff $\mathcal{O}_\mathsf{g}$ is unsatisfiable iff $\bot \in \mathsf{Ch}(\mathcal{P}_{\mathcal{O}_\mathsf{g}})$ iff $\bot \in \mathsf{Ch}(\mathcal{P}_{\mathcal{O}})$.

We conclude the section by showing that, given some ontology $\mathcal{O}$ with no more than n nominal variables per rule, the chase of $\mathcal{P}_{\mathcal{O}}$ can be computed in polynomial time (in the size of $\mathcal{O}$, holding n fixed). This is not a completely straightforward conclusion given that we do include function symbols in our definition of Datalog, and rules produced by the function γ may include rules with an arbitrarily large number of variables. Nevertheless, the following holds.

Lemma 7 *Fix an $\mathcal{ELV}^{++}$ signature and the Datalog signature S derived from it. For any $\mathcal{ELV}^{++}$ concept expression C with no more than n nominal variables, there is a set of rules $\mathsf{tr}(C)$ of polynomial size, containing fresh predicates, but no function symbols, and there is a fresh predicate C', such that for any Datalog program P not containing the fresh predicates of $\mathsf{tr}(C)$, for any substitution σ taking values in the terms of P's signature, and $\vec{v}$ being the nominal variables of C, if $\gamma_l(C, x)\sigma \subseteq \mathsf{Ch}(P)$ then $C'(x, \vec{v})\sigma \in \mathsf{Ch}(P \cup \mathsf{tr}(C))$, and if $C'(x, \vec{v})\sigma \in \mathsf{Ch}(P \cup \mathsf{tr}(C))$, then for some substitution σ' agreeing with σ on x and $\vec{v}$, $\gamma_l(C, x)\sigma' \subseteq \mathsf{Ch}(P)$.*

Proof By induction on the definition of γ_l. The most interesting case is $C = \exists R.D$ for a role R:

Let $\gamma_l(\exists R.D, x)\sigma \subseteq \mathsf{Ch}(P)$. So $R(x\sigma, w\sigma) \in \mathsf{Ch}(P)$, $\gamma_l(D, w)\sigma \subseteq \mathsf{Ch}(P)$. By inductive hypothesis, $D'(w, \vec{v}) \in \mathsf{Ch}(P \cup \mathsf{tr}(D))$, where $\vec{v}$ is the vector of nominal variables of D, which are also all those of C. So we let $\mathsf{tr}(C) = \mathsf{tr}(D) \cup \{R(x, w) \wedge D'(w, \vec{v}) \to C'(x, \vec{v})\}$. Now $C'(x\sigma, \vec{v}\sigma) \in \mathsf{Ch}(P \cup \mathsf{tr}(C))$, as

required. For the converse, let $C'(x\sigma, \vec{v}\sigma) \in \mathsf{Ch}(P \cup \mathsf{tr}(C))$. Since C' occurs in the head of just one rule in $\mathsf{tr}(C)$, and never in a rule body, it follows that $R(x\sigma, u) \wedge D'(u, \vec{v}\sigma) \in \mathsf{Ch}(P \cup \mathsf{tr}(D))$ for some u. Let σ'' be the substitution that is equal to σ except that $w\sigma'' = u$. Then by the inductive hypothesis, there is a substitution σ''' equal to σ'' on $\vec{v}$ and w such that $\gamma_l(D, w)\sigma''' \subseteq \mathsf{Ch}(P)$. Now x does not occur in $\gamma_l(D, w)$ by the fresh variable assumption, so let $\sigma' = \sigma''' = \sigma$ on $\vec{v}$ and $= \sigma$ on x, and $\gamma_l(D, x)\sigma' \subseteq \mathsf{Ch}(P)$ as required. Finally, assume as an inductive hypothesis that the size of $\mathsf{tr}(C)$ is bounded by a polynomial function p of the size of C, for all formulas of depth less than that of C, and that p has sufficiently large positive slope. We will prove that p bounds the size of $\mathsf{tr}(C)$ as well. Now $|\mathsf{tr}(C)| = |\mathsf{tr}(D)| + k$, k a constant depending on the fixed variable count n. $|C| = |D| + 3$. Without loss of generality assume the bounding polynomial p has slope at least $k/3$ everywhere. Then $|\mathsf{tr}(C)| = |\mathsf{tr}(D)| + k \leq p(|D|) + k \leq p(|D| + 3) = p(|C|)$, as required. Each other inductive case is similar, requiring some lower bound $s(n)$ on the slope of p, but we can assume without loss of generality that the slope of p is everywhere greater than all the $s(n)$. If we do so, it follows by induction that $p(|C|) \geq |\mathsf{tr}(C)|$ for all C. □

Theorem 3 *For each n, given an $\mathcal{EL}\mathcal{V}^{++}$ ontology $\mathcal{O}$ with no more than n nominal variables, we can compute $\mathsf{Ch}(\mathcal{P}_\mathcal{O})$ in* PTIME *with respect to the size of $\mathcal{O}$.*

Proof Let $\mathcal{O}_\mathsf{g}$ be the ontology described in Definition 6. Then, according to Lemma 6 we have that $\mathsf{Ch}(\mathcal{P}_{\mathcal{O}_\mathsf{g}}) = \mathsf{Ch}(\mathcal{P}_\mathcal{O})$. Therefore, it suffices to show that $\mathsf{Ch}(\mathcal{P}_{\mathcal{O}_\mathsf{g}})$ can be computed in polynomial time to show the theorem. By assumption, there are at most n different nominal schemas per axiom in $\mathcal{O}$ and hence, $\mathcal{O}_\mathsf{g}$ is polynomial in $\mathcal{O}$.

We can obtain a conservative extension $\mathcal{P}'_{\mathcal{O}_\mathsf{g}}$ of $\mathcal{P}_{\mathcal{O}_\mathsf{g}}$ in polynomial time in which rules contain a fixed number of variables, due to lemma 7, by replacing each $\gamma_l(C, t)$ in the definition of γ with $C'(t)$, using the predicate C' of that lemma, and adding the polynomial ruleset $\mathsf{tr}(C)$ to the program. Note that $\gamma_r(C, t)$ never contains more than one variable when C contains no nominal schemas. So now there are no rules left in $\mathcal{P}'_{\mathcal{O}_\mathsf{g}}$ with more than two variables except those derived from role-chain axioms, which can be conservatively normalized to rules containing 3 variables in the usual way. Moreover, note that only a polynomially large number of terms may occur in $\mathsf{Ch}(\mathcal{P}'_{\mathcal{O}_\mathsf{g}})$ since, for every term of the form $f(t_1, \ldots, t_k)$ in $\mathcal{P}'_{\mathcal{O}_\mathsf{g}}$ and every $i = 1, \ldots, k$, $t_i \in \mathbf{I}$. Therefore, the number of facts in $\mathsf{Ch}(\mathcal{P}'_{\mathcal{O}_\mathsf{g}})$ is also polynomially bounded, since the arity of predicates in $\mathcal{P}'_{\mathcal{O}_\mathsf{g}}$ is bounded. We hence conclude that the chase sequence of a program $\mathcal{P}'_{\mathcal{O}_\mathsf{g}}$ contains at most polynomially many elements and, since the rules in $\mathcal{P}'_{\mathcal{O}_\mathsf{g}}$ only contain a fixed amount of variables, every step of the chase can be computed in polynomial time. □

The above result indicates that the proposed reasoning approach is worst-case optimal in one sense, since assertion retrieval over $\mathcal{EL}^{++}$ ontologies is PTIME-hard with respect to logspace [10].

5 Evaluation

In this section we present some experiments in which we compare the performance of an implementation of the algorithm from Section 4 with that of other state-of-the-art reasoners. We first describe the ontologies used in our experiments (Section 5.1) and the reasoners considered (Section 5.2), and then present and discuss the results of our experiments (Section 5.3).

5.1 Ontologies

We consider three different ontologies in our evaluation; one benchmark and two real-world knowledge bases.

- *LUBM* is a widely-used benchmark [12] that models information about the academic domain. LUBM includes a set of TBox axioms and a data generator which can be used to generate ABox assertions for k universities where k is an arbitrary parameter passed on to the generator. For more information about this benchmark, see `http://swat.cse.lehigh.edu/projects/lubm/`.
- *Reactome* and *Uniprot* are real-world ontologies developed by the European Bioinformatics Institute (EBI) (see `https://www.ebi.ac.uk/`). These ontologies are particularly interesting, as their data is realistic and is used in a number of applications. Moreover, both ontologies contain an expressive set of TBox axioms and a very large amount of ABox assertions. For more information about these ontologies, see `https://reactome.org/` and `https://www.uniprot.org/`, respectively.

We preprocessed all of the above ontologies in the following manner: First, we remove all non-$\mathcal{ELV}^{++}$ axioms. Then, we add some hand-crafted, DL-safe rules to each ontology. We added DL-safe rules instead of axioms with nominal schemas as only the former can be processed by Konclude—one of the reasoners considered in the evaluation. Note that, DL-safe rules can be faithfully expressed using axioms with nominal schemas [7]. For instance, we can replace the DL-safe rule

$$\mathsf{Student}(x) \wedge \mathsf{takesCourse}(x, z) \wedge \mathsf{Course}(z) \wedge \mathsf{advisor}(x, y) \wedge \mathsf{Faculty}(y) \wedge \mathsf{teacherOf}(y, z) \rightarrow \mathsf{taughtBy}(x, y)$$

with the DL axiom

$$\{x\} \sqcap \mathsf{Student} \sqcap \exists \mathsf{takesCourse}.(\{z\} \sqcap \mathsf{Course}) \sqcap \exists \mathsf{advisor}.(\{y\} \sqcap \mathsf{Faculty} \sqcap \exists \mathsf{teacherOf}.\{z\}) \sqsubseteq \{x\} \sqcap \exists \mathsf{taughtBy}.\{y\}$$

preserving the outcome of all reasoning tasks.

The DL-safe rules added to LUBM, Reactome, and Uniprot are presented in Listings 5, 6, and 7, respectively. We have created the rules for LUBM ourselves. The rules for Reactome and Uniprot have been constructed from queries obtained in the evaluation of [43] (see `https://www.cs.ox.ac.uk/isg/tools/PAGOdA/2015/jair/` for more information).

$\mathsf{GradStudent}(x) \wedge \mathsf{UndergraduateDegreeFrom}(x, y) \wedge \mathsf{University}(y) \wedge$
$\mathsf{MemberOf}(x, z) \wedge \mathsf{Dept}(z) \wedge \mathsf{SubOrganizationOf}(z, y) \rightarrow \mathsf{MemberOf}(x, y)$

$\mathsf{Student}(x) \wedge \mathsf{TakesCourse}(x, y) \wedge \mathsf{Course}(y) \wedge \mathsf{TeacherOf}(z, y) \rightarrow \mathsf{WorksFor}(x, z)$

$\mathsf{Student}(x) \wedge \mathsf{MemberOf}(x, y) \wedge \mathsf{Dept}(y) \wedge \mathsf{SubOrganizationOf}(y, z) \rightarrow \mathsf{HasAlumnus}(x, z)$

$\mathsf{Student}(x) \wedge \mathsf{TakesCourse}(x, z) \wedge \mathsf{Course}(z) \wedge$
$\mathsf{Advisor}(x, y) \wedge \mathsf{Faculty}(y) \wedge \mathsf{TeacherOf}(y, z) \rightarrow \mathsf{GradStudent}(x) \wedge$

$\mathsf{Chair}(x) \wedge \mathsf{WorksFor}(x, y) \wedge \mathsf{Dept}(y) \wedge \mathsf{SubOrganizationOf}(y, z) \rightarrow \mathsf{WorksFor}(x, z)$

$\mathsf{Student}(x) \wedge \mathsf{TakesCourse}(x, z) \wedge \mathsf{Course}(z) \wedge \mathsf{MemberOf}(x, w) \wedge$
$\mathsf{TeacherOf}(y, z) \wedge \mathsf{Faculty}(y) \wedge \mathsf{WorksFor}(y, w) \wedge \mathsf{Dept}(w) \rightarrow \mathsf{MemberOf}(x, w)$

$\mathsf{Faculty}(x) \wedge \mathsf{DegreeFrom}(x, y) \wedge \mathsf{University}(y) \wedge \mathsf{MemberOf}(x, z) \wedge \mathsf{Dept}(z) \wedge$
$\mathsf{SubOrganizationOf}(z, y) \rightarrow \mathsf{AffiliateOf}(x, y)$

$\mathsf{PublicationAuthor}(w, z) \wedge \mathsf{Professor}(z) \wedge \mathsf{MemberOf}(z, x) \wedge \mathsf{Dept}(x) \wedge$
$\mathsf{PublicationAuthor}(w, v) \wedge \mathsf{Professor}(v) \wedge \mathsf{MemberOf}(v, y) \wedge \mathsf{Dept}(y) \rightarrow \mathsf{MemberOf}(w, y)$

Listing 5 DL-Safe Rules added to LUBM.

$\mathsf{Pathway}(x) \wedge \mathsf{PathwayComponent}(x, y) \wedge \mathsf{BiochemicalReaction}(y) \wedge \mathsf{Participant}(y, z) \wedge$
$\mathsf{Complex}(z) \rightarrow \mathsf{R1}(x)$

$\mathsf{Pathway}(x) \wedge \mathsf{PathwayComponent}(x, y) \wedge \mathsf{BiochemicalReaction}(y) \wedge \mathsf{Participant}(y, z) \wedge$
$\mathsf{Protein}(z) \wedge \mathsf{entityReference}(z, w) \rightarrow \mathsf{R2}(x)$

$\mathsf{Stoichiometry}(x) \wedge \mathsf{PhysicalEntity}(x, y) \wedge \mathsf{PhysicalEntity}(y) \wedge \mathsf{cellularLocation}(y, z) \wedge$
$\mathsf{CellularLocationVocabulary}(z) \rightarrow \mathsf{R3}(x)$

$\mathsf{Pathway}(x) \wedge \mathsf{PathwayComponent}(x, y) \wedge \mathsf{Participant}(y, z) \wedge \mathsf{Protein}(z) \wedge$
$\mathsf{PathwayComponent}(x, w) \wedge \mathsf{Participant}(w, z) \rightarrow \mathsf{R4}(x)$

$\mathsf{FeatureLocation}(x, y) \wedge \mathsf{SequenceIntervalBegin}(y, z) \wedge \mathsf{SequenceSite}(z) \wedge$
$\mathsf{FeatureLocation}(x, w) \wedge \mathsf{SequenceIntervalBegin}(w, z) \rightarrow \mathsf{R5}(x)$

$\mathsf{ParticipantStoichiometry}(x, y) \wedge \mathsf{PhysicalEntity}(y, z) \wedge \mathsf{ParticipantStoichiometry}(x, w) \wedge$
$\mathsf{PhysicalEntity}(w, z) \rightarrow \mathsf{R6}(x)$

Listing 6 DL-Safe Rules added to Reactome.

5.2 Reasoners

We consider three different reasoners in our evaluation, which we describe in detail across the following paragraphs.

– *ELVLog* is our proprietary implementation of the reasoning algorithm discussed in Section 4. Roughly, this implementation works in the following manner: First, we use the OWLAPI [16] to process and transform $\mathcal{EL}^{++}$ ontologies into Datalog programs as described in Definition 7. Then, we use the rule engine VLog [40,41] to compute the chase of the result-

$$\mathsf{Protein}(x) \wedge \mathsf{annotation}(x, y) \wedge \mathsf{TransmembraneAnnotation}(y) \wedge \mathsf{Range}(y, z) \rightarrow \mathsf{R1}(x)$$

$$\mathsf{TransmembraneAnnotation}(x) \wedge \mathsf{range}(x, y) \wedge \mathsf{range}(x, z) \rightarrow \mathsf{R2}(y, z)$$

$$\mathsf{Protein}(x) \wedge \mathsf{organism}(x, y) \wedge \mathsf{annotation}(x, z) \rightarrow \mathsf{R3}(x, z)$$

$$\mathsf{locatedIn}(x, w) \wedge \mathsf{cellularComponent}(w, z) \wedge \mathsf{cellularComponent}(z, y) \wedge \mathsf{CellularComponent}(y) \rightarrow \mathsf{R4}(x, z)$$

$$\mathsf{source}(x, y) \wedge \mathsf{source}(w, y) \wedge \mathsf{database}(y, z) \rightarrow \mathsf{R5}(x, w)$$

$$\mathsf{attribution}(w, x) \wedge \mathsf{source}(x, y) \wedge \mathsf{database}(y, z) \rightarrow \mathsf{R6}(x, y)$$

$$\mathsf{translatedFrom}(w, x) \wedge \mathsf{locatedOn}(x, y) \wedge \mathsf{database}(x, z) \rightarrow \mathsf{R7}(x, y)$$

Listing 7 DL-Safe Rules added to Uniprot.

ing programs. This implementation is available at `https://github.com/dcarralma/ELVLog`.

- *Konclude* [39] is a full-fledged OWL 2 reasoner. It is implemented in C++ and uses a reasoning technique that is based on a highly optimized tableau algorithm assisted by a completion-based saturation procedure. Konclude has proven itself as one of the most efficient OWL 2 reasoners in the Ontology Reasoning Evaluation (ORE) Workshop 2015 [35]. Moreover, Konclude has been shown to significantly outperform other existing tools such as Hermit 1.3.7 [34] and Pellet [36] when it comes to solve reasoning tasks over ontologies featuring DL-safe rules [38]. For more information, see `http://derivo.de/produkte/konclude/`.
- *ELK* [20] is a profile-specific reasoner for the OWL 2 EL profile, which is implemented in Java. As Konclude, ELK has proven to be quite competitive at the ORE competition in 2015 [35]. Unfortunately, ELK supports neither DL-safe rules nor nominal schemas. Nevertheless, we include it in our evaluation, to verify that ELVLog not only outperforms full-fledged DL reasoners, such as Konclude, but is also competitive when compared against this profile-specific tool. For more information, see `https://github.com/liveontologies/elk-reasoner`.

5.3 Results

We conduct two different experiments in which we compare ELVLog to Konclude and ELK, separately.

- *ELVLog vs Konclude*: in this experiment, we task Konclude and ELVLog with solving assertion retrieval over LUBM, Reactome, and Uniprot. Specifically, we used the Realization service provided by Konclude. Recall that assertion retrieval means computing *all* of the assertions entailed by a knowledge base - that is, an input ontology together with input set of assertions. Note that the ontologies we used were supplemented with the DL-safe rules presented in Tables 5, 6, and 7. To evaluate the scalability

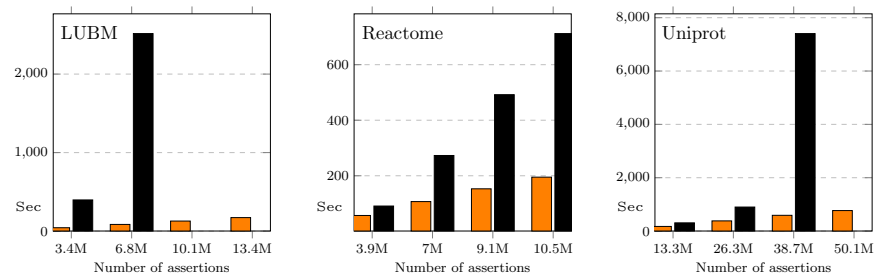

Fig. 8 Solving Assertion Retrieval with ELVLog (orange) and Konclude (black); Konclude did not finish LUBM with 10.1M assertions and Uniprot with 50.1M in less than 8 hours

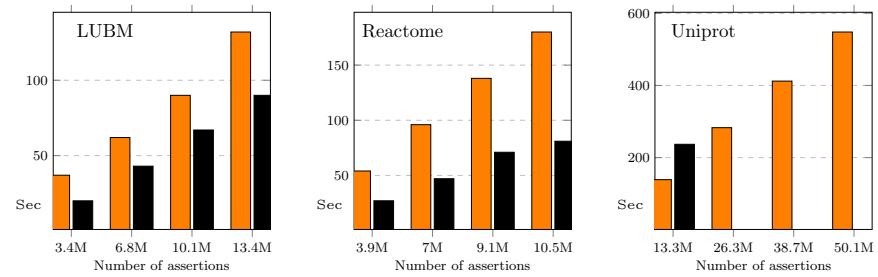

Fig. 9 Solving Assertion Retrieval with ELVLog (orange) and ELK (black); ELK runs out of memory when reasoning over Uniprot with 26.3M assertions

of both tools, we varied the size of the input assertion set for each of the three ontologies. For LUBM, sets of assertions of different sizes were generated using the data generator of the LUBM benchmark. For Reactome and Uniprot, we obtained these employing a data sampling algorithm [31].

– *ELVLog vs ELK*: this experiment is analogous to the previous one, but using ELK instead of Konclude. Note that the considered ontologies in this case *do not contain DL-safe rules* since ELK cannot deal with this type of OWL axiom.

The results for both experiments are presented in Tables 8 and 9, respectively. All experiments were performed on a MacBook Pro with 16GB of RAM and a 2,2 GHz Intel Core i7 processor. We set up a time-out for the experiments of 6 hours—that is, 21600 seconds. In the above tables, we indicate time-outs and out of memory errors by not including a bar for the corresponding ontology, ABox, and tool (for instance, when running LUBM with 10.1M assertions and Konclude, we obtained one such error). All of the files used in these experiments are available at `https://github.com/dcarralma/ELVLog`.

Because the considered ontologies contain a relatively small amount of TBox axioms, the time spent computing the corresponding Datalog programs by ELVLog is very low (for all cases, it took less than 2 seconds). Note, that

the results in Tables 8 and 9 for ELVLog include the time taken by this transformation.

The experiment presented in Table 8 shows that ELVLog is significantly more efficient than Konclude; in many cases, our implementation is up to an order of magnitude faster than Konclude. We believe that this is because VLog is way more efficient than Konclude in the presence of large amounts of data; an insight that has been confirmed by previous experiments [5,6,11].

Even though ELK is somewhat more efficient than ELVLog for both LUBM and Reactome, the latter tool can solve assertion retrieval for Uniprot with up to 50.1M assertions in less than 10 minutes, whilst the former runs out of memory with only 26.3M assertions (see Table 9). Therefore, we believe that ELVLog can also be a useful tool when it comes to reasoning with $\mathcal{EL}^{++}$ ontologies (i.e., without DL-safe rules or nominal schemas).

6 Conclusions and Future Work

We define and prove correctness of a worst-case optimal procedure for reasoning over $\mathcal{EL}^{++}$ ontologies extended with nominal schemas. We implement this algorithm and show that it is quite efficient: it not only improves upon the OWL 2 reasoner Konclude, but furthermore shows better performance in some cases than the OWL EL profile reasoner ELK.

As for future work, we intend to extend our approach to more expressive DL languages. More specifically, we intend to extend the algorithms presented in [5] and [6] to develop efficient algorithms for the DL languages Horn-$\mathcal{SRIQ}$ and Horn-$\mathcal{ALCHOIQ}$, respectively, that can deal with nominal schemas.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, second edn. (2007)
3. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Kaelbling, L.P., Saffiotti, A. (eds.) Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005). pp. 364–369. Professional Book Center (2005)
4. Baader, F., Lutz, C., Brandt, S.: Pushing the $\mathcal{EL}$ envelope further. In: Clark, K., Patel-Schneider, P. (eds.) Proc. of the 4th OWLED Workshop on OWL: Experiences and Directions. CEUR Workshop Proceedings, vol. 496 (2008)
5. Carral, D., Dragoste, I., Krötzsch, M.: The combined approach to query answering in Horn-$\mathcal{ALCHOIQ}$. In: Thielscher, M., Toni, F., Wolter, F. (eds.) Proc. of the 16th International Conference on Principles of Knowledge Representation and Reasoning (KR 2018). pp. 339–348. AAAI Press (2018)
6. Carral, D., González, L., Koopmann, P.: From Horn-$\mathcal{SRIQ}$ to Datalog: A data-independent transformation that preserves assertion entailment. In: Proc. of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019). pp. 2736–2743. AAAI Press (2019)
7. Carral, D., Hitzler, P.: Extending description logic rules. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, Ó., Presutti, V. (eds.) Proc. of the 9th Extended Semantic Web Conference (ESWC 2012). Lecture Notes in Computer Science, vol. 7295, pp. 345–359. Springer (2012)

8. Carral, D., Krisnadhi, A., Maier, F., Sengupta, K., Hitzler, P.: Reconciling OWL and rules. Tech. rep., Wright State University, Dayton, Ohio, U.S.A. (2011), available from `http://www.pascal-hitzler.de/` under the "Publications" tab
9. Carral, D., Wang, C., Hitzler, P.: Towards an efficient algorithm to reason over Description Logics extended with nominal schemas. In: Faber, W., Lembo, D. (eds.) Proc. of the 7th International Conference on Web Reasoning and Rule Systems (RR 2013). Lecture Notes in Computer Science, vol. 7994, pp. 65–79. Springer (2013)
10. Cuenca Grau, B.: Owl 2 web ontology language tractable fragments (second edition). Available at: `https://www.w3.org/2007/OWL/wiki/Tractable_Fragments` (2021/07/10)
11. Feier, C., Carral, D., Stefanoni, G., Grau, B.C., Horrocks, I.: The combined approach to query answering beyond the OWL 2 profiles. In: Yang, Q., J. Wooldridge, M. (eds.) Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015). pp. 2971–2977. AAAI Press (2015)
12. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. J. of Web Semantics 3(2-3), 158–182 (2005)
13. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer (Second Edition). W3C Recommendation (11 December 2012), available at `http://www.w3.org/TR/owl2-primer/`
14. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
15. Hitzler, P., Parsia, B.: Ontologies and rules. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, pp. 111–132. International Handbooks on Information Systems, Springer (2009)
16. Horridge, M., Bechhofer, S.: The OWL API: A java API for OWL ontologies. J. of Semantic Web 2(1), 11–21 (2011)
17. Horrocks, I., Patel-Schneider, P., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language. W3C Member Submission (21 May 2004), see `http://www.w3.org/Submission/SWRL/`
18. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006). pp. 57–67. AAAI Press (2006)
19. Kazakov, Y.: Saturation-Based Decision Procedures for Extensions of the Guarded Fragment. Ph.D. thesis, Universität des Saarlandes, Saarbrücken, Germany (March 2006)
20. Kazakov, Y., Krötzsch, M., Simancik, F.: The incredible ELK - from polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies. J. of Automated Reasoning 53(1), 1–61 (2014)
21. Kifer, M., Boley, H. (eds.): RIF Overview (Second Edition). W3C Working Group Note (2013), available at `http://www.w3.org/TR/rif-overview/`
22. Knorr, M., Carral, D., Hitzler, P., Krisnadhi, A., Maier, F., Wang, C.: Recent advances in integrating OWL and rules (technical communication). In: Krötzsch, M., Straccia, U. (eds.) Proc. of the 6th International Conference on Web Reasoning and Rule Systems (RR 2012). Lecture Notes in Computer Science, vol. 7497, pp. 225–228. Springer, Heidelberg (2012)
23. Knorr, M., Hitzler, P., Maier, F.: Reconciling OWL and non-monotonic rules for the Semantic Web. In: De Raedt, L., Bessiere, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., Lucas, P. (eds.) Proc. of the 20th European Conference on Artificial Intelligence (ECAI 2012). Frontiers in Artificial Intelligence and Applications, vol. 242, pp. 474–479. IOS Press, Amsterdam (2012)
24. Krisnadhi, A., Hitzler, P.: A tableau algorithm for description logics with nominal schema. In: Krötzsch, M., Straccia, U. (eds.) Proc. of the 6th International Conference on Web Reasoning and Rule Systems (RR 2012). Lecture Notes in Computer Science, vol. 7497, pp. 234–237. Springer (2012)
25. Krisnadhi, A., Maier, F., Hitzler, P.: OWL and rules. In: Polleres, A., d'Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P. (eds.) Reasoning Web. Semantic Technologies for the Web of Data. 7th International Summer School 2011, Tutorial Lectures. Lecture Notes in Computer Science, vol. 6848, pp. 382–415. Springer, Heidelberg (2011)
26. Krötzsch, M.: Description Logic Rules, Studies on the Semantic Web, vol. 008. IOS Press/AKA (2010)

27. Krötzsch, M., Rudolph, S., Hitzler, P.: ELP: Tractable rules for OWL 2. In: Sheth, A., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) Proc. of the 7th International Semantic Web Conference (ISWC 2008). Lecture Notes in Computer Science, vol. 5318, pp. 649–664. Springer (2008)
28. Krötzsch, M.: Efficient rule-based inferencing for OWL EL. In: Walsh, T. (ed.) Proc. of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011). AAAI Press/IJCAI (2011), 2668–2673
29. Krötzsch, M., Maier, F., Krisnadhi, A., Hitzler, P.: A better uncle for OWL: Nominal schemas for integrating rules and ontologies. In: Proc. of the 20th International Conference on World Wide Web (WWW 2011). pp. 645–654. ACM (2011)
30. Krötzsch, M., Rudolph, S.: Nominal schemas in Description Logics: Complexities clarified. In: Baral, C., De Giacomo, G., Eiter, T. (eds.) Proc. of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR 2014). AAAI Press (2014)
31. Leskovec, J., Faloutsos, C.: Sampling from large graphs. In: Eliassi-Rad, T., Ungar, L.H., Craven, M., Gunopulos, D. (eds.) Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 631–636. ACM (2006)
32. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Language: Profiles. W3C Recommendation (2009), available at `http://www.w3.org/TR/owl2-profiles/`
33. Motik, B., Sattler, U., Studer, R.: Query answering for OWL DL with rules. J. of Web Semantics 3(1), 41–60 (2005)
34. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for Description Logics. J. of Artificial Intelligence Research 36(1), 165–228 (2009)
35. Parsia, B., Matentzoglu, N., Gonçalves, R.S., Glimm, B., Steigmiller, A.: The OWL reasoner evaluation (ORE) 2015 competition report. J. of Automated Reasoning 59(4), 455–482 (2017)
36. Sirin, E., Parsia, B., Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL DL reasoner. J. of Web Semantics 5(2), 51–53 (2007)
37. Steigmiller, A., Glimm, B., Liebig, T.: Nominal schema absorption. In: Rossi, F. (ed.) Proc. of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013). IJCAI/AAAI (2013)
38. Steigmiller, A., Glimm, B., Liebig, T.: Reasoning with nominal schemas through absorption. J. of Automated Reasoning 53(4), 351–405 (2014)
39. Steigmiller, A., Liebig, T., Glimm, B.: Konclude: System description. J. of Web Semantics 27, 78–85 (2014)
40. Urbani, J., Jacobs, C.J.H., Krötzsch, M.: Column-oriented datalog materialization for large knowledge graphs. In: Schuurmans, D., Wellman, M.P. (eds.) Proc. of the 30th AAAI Conference on Artificial Intelligence (AAAI 2016). pp. 258–264. AAAI Press (2016)
41. Urbani, J., Krötzsch, M., Jacobs, C.J.H., Dragoste, I., Carral, D.: Efficient model construction for horn logic with vlog – system description. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) Proc. of the 9th International Joint Conference on Automated Reasoning (IJCAR 2018) Held as Part of the Federated Logic Conference (FloC 2018). Lecture Notes in Computer Science, vol. 10900, pp. 680–688. Springer (2018)
42. Wang, C., Hitzler, P.: A resolution procedure for Description Logics with nominal schemas. In: Takeda, H., Giu, Y., Mizoguchi, R., Kitamura, Y. (eds.) Proc. of the 2nd Joint International Conference on Semantic Technology (JIST 2012). Lecture Notes in Computer Science, vol. 7774, pp. 1–16. Springer, Heidelberg (2012)
43. Zhou, Y., Cuenca Grau, B., Nenov, Y., Kaminski, M., Horrocks, I.: PAGOdA: Pay-as-you-go ontology query answering using a Datalog reasoner. J. of Artificial Intelligence Research 54, 309–367 (2015)