Reasoning with the Core Chase: the Case of SHACL Validation over ELHI Knowledge Bases

Anouk Oudshoorn[1], Magdalena Ortiz[1] and Mantas Šimkus[1]

[1] *Institute of Logic and Computation, TU Wien, Austria*

Abstract

Many approaches to query answering in Description Logics (DLs) adopt the certain answer semantics, which is natural for standard query languages without negation, but falls short when negation is involved. Since negation plays a crucial role in many applications, alternative semantics are needed. *Core universal models* are increasingly accepted as a means to give a semantics to queries with negation. In this paper, we consider the problem of evaluating (possibly recursive) SHACL expressions over the core universal model of an $\mathcal{ELHI}$ KB. SHACL, recently proposed as a standard constraint language for RDF data, is a natural and powerful query language, and the fragment we consider adds regular path navigation to semi-positive monadic Datalog with acyclic rule bodies. We first propose a construction of a finite representation of the core universal model, based on a novel calculus for $\mathcal{ELHI}$ that gives up the data-independence to avoid the best-case exponential behaviour of similar approaches, and which we believe can lead towards an efficient implementation. Then we leverage this finite representation to reduce validation in the presence of ontologies to plain SHACL validation, similarly to previous algorithms but avoiding their best-case exponential behaviour. Our algorithms yield tight data- and combined-complexity bounds for the studied SHACL validation problem, which coincide with plain consistency testing in $\mathcal{ELHI}$.

Keywords

core chase, query rewriting, SHACL, ELHI, Horn DLs

1. Introduction

Evaluating queries over description logic (DL) knowledge bases has been the subject of huge research efforts over almost two decades. Here, the *certain answer semantics* has played a prominent role in elegantly handling information incompleteness for large classes of queries: answering a query consists of computing the intersection of query answers over all models of the input knowledge base (KB). This problem can easily become computationally very costly for expressive DLs, and it is provably more expensive than standard reasoning in most of the popular extensions of $\mathcal{ALC}$ [1]. But fortunately, this computational hurdle can usually be avoided in the so-called *Horn* DLs, which disallow ontological statements that involve disjunction, leading in turn to the *universal model property* [2, 3]. This property guarantees that a consistent KB always has a model that represents all models (technically, that can be homomorphically mapped into any model) and is thus sufficient to compute the certain answers to all conjunctive queries.

DL 2024: 37th International Workshop on Description Logics, June 18–21, 2024, Bergen, Norway

anouk.oudshoorn@tuwien.ac.at (A. Oudshoorn); magdalena.ortiz@tuwien.ac.at (M. Ortiz); mantas.simkus@tuwien.ac.at (M. Šimkus)

0009-0006-4638-5948 (A. Oudshoorn); 0000-0002-2344-9658 (M. Ortiz); 0000-0003-0632-0294 (M. Šimkus)

The universal model property is closely related to the notion of a *chase procedure* known from databases, originally studied for reasoning about integrity constraints [4]. There are several chase variants with different properties (e.g., *oblivious*, *Skolem*, *core*) [5, 6, 7], and all of them can be used to obtain a model that is universal in the sense above. Query answering is still far from trivial even when universal models exists. For standard DLs, the chase does not terminate in general and universal models can be infinite, so we need to reason about them without building them explicitly. Nevertheless, the universal model property of Horn DLs has enabled the development of several algorithms and practical implementations of query answering over large classes of *positive queries*, see e.g., [2, 3, 8, 9, 10, 11] and their references.

Unfortunately, the situation is very different when queries involve *negation as failure*, which allows us to query about the absence of information, e.g., to ask for objects that do not (provably) have some property. Such queries are highly desirable in KR and have been advocated in several papers [12, 13, 14, 15]. However, they are not closed under homomorphisms, which means that different universal models may lead to different answers. Apparently innocuous differences in the chase procedure may affect the query answers, and the certain answer semantics becomes inadequate; see, e.g., the discussion in [14] and [15]. The *core universal model* is increasingly accepted as a way to define the semantics of these queries [14, 15], since it can be seen as the structure that best represents the intended meaning of the given input data and the terminological axioms, without any superfluous structures or redundant elements.

The first major contribution of this paper is a method for computing a finite representation of a (possibly infinite)core universal model of an $\mathcal{ELHI}$ KB. This method includes a *data-aware* version of a consequence-based calculus for $\mathcal{ELHI}$. Specifically, we define a collection of reasoning rules that incrementally builds a finite representation of a core universal model for a given input ABox. Unlike previous consequence-based inference procedures for $\mathcal{ELHI}$, are the rules in our calculus aware of input data. This avoids the main drawback of the previous approaches: data-independent computations easily lead to best-case exponential behavior.

After presenting our calculus, we use it to address a timely and challenging problem: validation of (possibly recursive) SHACL in the presence of ontologies. SHACL is a recently introduced W3C recommendation for writing constraints over RDF data [16]. It has also been formalised in logic and has connections to DLs [17, 18, 19, 20]. The SHACL standard envisions SHACL validation performed in the presence of ontologies but does not describe how this should be realised. SHACL is closely related to DLs and typical formalisms studied in the setting of ontology-mediated queries. We concentrate on *semi-positive* SHACL shapes, which allow for negation of ontology predicates, but not of *shape names*. This fragment can be seen as semi-positive monadic Datalog queries with acyclic rule bodies and regular path expressions, and it already raises the issues with the certain answer semantics discussed above.

Following our previous work [15], we use a core universal model of the data and ontology for validation. This leads us to our second contribution: an algorithm for evaluating SHACL expressions over the finite representation of a core universal model of an $\mathcal{ELHI}$ KB. This procedure is based on the rewriting from [15], but unlike that work, the algorithm presented here is not best-case exponential, potentially paving the way towards implementation. Moreover, we show that checking if a given $\mathcal{ELHI}$ KB validates a shapes graph is ExpTime-complete and PTime-complete in combined and data complexity, respectively. These bounds are the best we could hope for, since they coincide with the complexity of consistency testing in $\mathcal{ELHI}$.

2. Preliminaries

Let N_C, N_I, and N_R be infinite sets of concept names, individual names, and role names, respectively. Let $N_C^+ := N_C \cup \{\top\}$ and $N_R^+ := \{p, p^- \mid p \in N_R\}$ denote *roles*. For every $p \in N_R$, let $(p^-)^- = p$. For each set of roles $R \subseteq N_R^+$, set $R^- := \{r^- \mid r \in N_R^+\}$. Let N_B be an infinite set of blank nodes disjoint from N_I. Call $A(c)$ and $r(c,d)$ *atoms*, for each $A \in N_C$, $r \in N_R^+$ and $\{c,d\} \subseteq N_I \cup N_B$. We let $\Delta^{\mathcal{A}}$ denote the *domain* of $\mathcal{A}$, i.e. the elements of $N_I \cup N_B$ that appear in $\mathcal{A}$. In case $\Delta^{\mathcal{A}} \subseteq N_I$, we call $\mathcal{A}$ an *ABox*. In the rest of this paper, we assume that for all ABoxes $\mathcal{A}$, if $r(a,b) \in \mathcal{A}$, then $r^-(b,a) \in \mathcal{A}$ too. For a tuple $\vec{x} = (x_1, \ldots, x_n)$ and $1 \leq j \leq n$, we let $\pi_i(\vec{x}) = x_i$ be its *i-th projection.*

Morphisms. Let $\mathcal{A}$ and $\mathcal{A}'$ be sets of atoms. A *homomorphism* from $\mathcal{A}$ to $\mathcal{A}'$ is a function $h : \Delta^{\mathcal{A}} \to \Delta^{\mathcal{A}'}$ such that for all $\{c,d\} \subseteq N_I \cup N_B$, all $A \in N_C$ and all $p \in N_R$, (i) if $c \in \Delta^{\mathcal{A}} \cap N_I$, then $h(c) = c$, (ii) if $A(c) \in \mathcal{A}$, then $A(h(c)) \in \mathcal{A}'$, and (iii) if $\{p(c,d), p^-(d,c)\} \cap \mathcal{A} \neq \{\}$, then $\{p(h(c),h(d)), p^-(h(d),h(c))\} \cap \mathcal{A}' \neq \{\}$. A homomorphism is called *strong* when (ii) and (iii) are strengthened to "$A(c) \in \mathcal{A}$ iff $A(h(c)) \in \mathcal{A}'$" and "$\{p(c,d), p^-(d,c)\} \cap \mathcal{A} \neq \{\}$ iff $\{p(h(c),h(d)), p^-(h(d),h(c))\} \cap \mathcal{A}' \neq \{\}$", respectively. An *embedding* is a strong injective homomorphism, an *isomorphism* is a surjective embedding and an *endomorphism* of $\mathcal{A}$ is a homomorphism from $\mathcal{A}$ to itself. A set of atoms is a *core* when all its endomorphisms are embeddings. The core of a set of atoms $\mathcal{A}$ is a set of atoms $\mathcal{B} \subseteq \mathcal{A}$, such that (i) there exists an endomorphism h from $\mathcal{A}$ to $\mathcal{B}$, (ii) $\mathcal{B}$ is the restriction to the image of h, and (iii) $\mathcal{B}$ is a core.We write $\mathcal{A} \xrightarrow{core} \mathcal{B}$. Each finite set of atoms has a core that is unique up to isomorphism [21].

The Description logic $\mathcal{ELHI}$. We are considering a normalised version of $\mathcal{ELHI}$, i.e., the axioms we consider have one of the following normal forms.

$$A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \quad A_1 \sqsubseteq \exists r.A_2 \quad \exists r.A_1 \sqsubseteq B \quad r \sqsubseteq r'$$

Here, $\{A_1, \ldots, A_n\} \subseteq N_C^+$, $B \in N_C^+ \cup \{\bot\}$ and $\{r, r'\} \subseteq N_R^+$. An $\mathcal{ELHI}$ TBox $\mathcal{T}$ is a set of such axioms. An ($\mathcal{ELHI}$) knowledge base $(\mathcal{T}, \mathcal{A})$ consists of an ABox $\mathcal{A}$ and an $\mathcal{ELHI}$ TBox $\mathcal{T}$. As usual, we use first-order interpretations to define the semantics. Note that any set of atoms $\mathcal{A}$ can be viewed as an interpretation with domain $\Delta^{\mathcal{A}}$.

Building the chase. The function f_x, for each $x \in N_I \cup N_B$, that translates concepts into a set of atoms is inductively defined in the following way: $f_x(\top) := \{\}$, $f_x(A) := \{A(x)\}$, $f_x(\exists r.A) := \{r(x,y), r^-(y,x), A(y)\}$ for some fresh variable y and $f_x(C \sqcap C') := f_x(C) \cup f_x(C')$. We say c is an *applicable match* for an axiom $C \sqsubseteq D \in \mathcal{T}$ in a set of atoms $\mathcal{A}$, when there exists a homomorphism h from $f_x(C)$ to $\mathcal{A}$, such that $h(x) = c$ and there is no homomorphism from $f_x(D)$ to $\mathcal{A}$ such that $h(x) = c$. We use the notation $(c, C \sqsubseteq D) \in m_\exists(\mathcal{T}, \mathcal{A})$ when D is of the form $\exists r.A$ for some $r \in N_R^+$, $A \in N_C^+$, and the notation $(c, C \sqsubseteq D) \in m(\mathcal{T}, \mathcal{A})$ when $D \in N_C$. Similarly, (c,d) is an applicable match for $r \sqsubseteq r' \in \mathcal{T}$ in a set of atoms $\mathcal{A}$, when $r(c,d) \in \mathcal{A}$ and $r'(c,d) \notin \mathcal{A}$. In that case, we write $((c,d), r \sqsubseteq r') \in m(\mathcal{T}, \mathcal{A})$.

Definition 1. *For an $\mathcal{ELHI}$ TBox $\mathcal{T}$, a set $I \subseteq N_I \cup N_B$ and ABoxes $\mathcal{A}, \mathcal{A}'$, we let $\mathcal{A} \xrightarrow{\mathcal{T},I} \mathcal{A}'$ if*

$$\mathcal{A}' = \mathcal{A} \cup \bigcup_{(c, C \sqsubseteq D) \in m_\exists(\mathcal{T},\mathcal{A}), c \in I} f_c(D) \cup \bigcup_{(c, C \sqsubseteq D) \in m(\mathcal{T},\mathcal{A})} f_c(D) \cup \bigcup_{((c,d), r \sqsubseteq r') \in m(\mathcal{T},\mathcal{A})} r'(c,d).$$

When $I = N_I \cup N_B$, we drop the I.

Note that the I restricts the set of nodes for which we consider applicable existential matches. We use the notation $\mathcal{A}(\rightarrow)^{\omega}\mathcal{A}'$ to denote that there exists an n such that $\mathcal{A}(\rightarrow)^{n}\mathcal{A}'$ and for all $\mathcal{A}''$ such that $\mathcal{A}' \rightarrow \mathcal{A}''$ we find $\mathcal{A}' = \mathcal{A}''$, for each binary relation $\rightarrow$. With $\circ$ we denote the concatenation of two binary relations, that is, $\mathcal{A} \rightarrow \circ \rightarrow' \mathcal{B}$ iff there exists $\mathcal{A}'$ such that $\mathcal{A} \rightarrow \mathcal{A}'$ and $\mathcal{A}' \rightarrow' \mathcal{B}$, for each pair of binary relations $\rightarrow$ and $\rightarrow'$.

Core chase. In the *core chase*, all applicable matches are fired simultaneously, followed by a core check. This procedure is repeated until it terminates (which is not guaranteed): given a knowledge base $\mathcal{T}, \mathcal{A}$, the core chase is the unique, up to isomorphism, structure $\mathcal{B}$ such that $\mathcal{A}(\xrightarrow{\mathcal{T}} \circ \xrightarrow{core})^{\omega}\mathcal{B}$ [7]. This procedure finds a *finite* universal (core) model whenever it exists, but it does not specify how to create an *infinite* structure from a series of finite chase structures: simply taking the union of $\mathcal{B}_i$, such that $\mathcal{A}(\xrightarrow{\mathcal{T}} \circ \xrightarrow{core})^{i}\mathcal{B}_i$, does in general not produce a core. Take for instance $\mathcal{A} = \{A(c)\}$ and $\mathcal{T} = \{A \sqsubseteq \exists r.A, A \sqsubseteq \exists s.A, r \sqsubseteq s\}$. The core chase is generalised to infinite structures in [22] by the so-called *stable chase*. The *austere universal model*, introduced in [15] for *DL-Lite*$_{\mathcal{R}}$, is constructed using a *good successor configuration* that ensures that the axioms are locally satisfied while preserving conditions of a core at each step. Here we follow a similar approach, and moreover, we prove that these local conditions are enough to obtain the core chase (whenever the procedure terminates).

3. Good Successor Configuration

In this section, we present our first major contribution: computing a finite representation of a core universal model for a given $\mathcal{ELHI}$ KB. In a nutshell, the good successor configuration tells us for each configuration of a node and its immediate 'neighbourhood', the precise configuration of blank nodes that should be introduced as immediate successors.

In contrast to *DL-Lite*$_{\mathcal{R}}$, in $\mathcal{ELHI}$ the relevant context is not only the incoming roles r, but also the concepts satisfied at the already existing neighbours. For simplicity, we focus on describing the good successors of the blank nodes introduced during the chase, which have a unique predecessor in the forest-like structure. For describing the successor configurations, we use *types* and *successor types*. A *type* $t \in \mathcal{F}$ is defined as an element of $\mathcal{P}(N_C^+) \times \mathcal{P}(N_R^+) \times \mathcal{P}(N_C^+)$, and a *successor type* $u \in \mathcal{S}$ is an element of $\mathcal{P}(N_R^+) \times \mathcal{P}(N_C^+)$. A type t describes a pair of nodes and the roles between them, while a successor type describes a set of roles leading to one node. A pair $(t, \{u_1, \ldots, u_n\})$ in the good configuration means that a node that is connected to its predecessor as described by t, and needs successors as prescribed by $u_1, \cdots, u_n$, see Figure 1. Furthermore, we define the *inverse* function $inv : \mathcal{P}(N_R^+) \times \mathcal{P}(N_C^+) \rightarrow \mathcal{P}(N_C^+) \times \mathcal{P}(N_R^+)$ on successor types by setting $inv(u) := (\pi_2(u), (\pi_1(u))^-)$. (Recall that π_1 and π_2 are the first and second projection of a tuple, respectively, as defined in the preliminaries.)

Defining the successors of all types that may occur in the chase for any possible ABox would necessarily lead to a strictly exponential behaviour. While exponentiality is in general unavoidable, as standard reasoning tasks are ExpTime complete, we want to avoid it for as many instances as possible. Hence in the following we build the good successor configuration carefully, to avoid best-case exponential behaviour. We start from the input ABox $\mathcal{A}$, and extend it to satisfy all applicable axioms, but we restrict the satisfaction of existential axioms to only create blank nodes that are directly neighbouring the ABox. We call this structure $\mathcal{A}_1$. The freshly introduced blank nodes in $\mathcal{A}_1$ have exactly one predecessor and can thus be described by a type. We take exactly those types as initial *live types*, and compute the *pre-good* configuration of successors that each live type needs. In each step of the computation we add to the live types the new successor types. It might happen that we encounter a so-called *loop computation*: the created successors may propagate some concept B back to the current node, This information is saved in a relation L as the pair of the type and the implied concept, and the pre-good successor pairs are updated with this information. Unlike *DL-Lite*$_\mathcal{R}$, derived information in $\mathcal{ELHI}$can propagate through the model, and in particular, affect the concepts satisfied by ABox individuals. Thus the information from the loop computation L must be used to update also the extended ABox, from which we retrieve fresh, updated live types for the next round of building pre-good successors. This continues in rounds until the process terminates.

Definition 2. *Given an $\mathcal{ELHI}$ knowledge base $(\mathcal{T}, \mathcal{A})$. Let $\mathcal{A}_1$ be the unique (up to isomorphism) set of atoms such that $\mathcal{A}(\xrightarrow{\mathcal{T},N_I})^\omega \circ \xrightarrow{core} \mathcal{A}_1$. Let F_i be the following set of types, for each $i \geq 1$*

$$\begin{aligned} F_i := \{t \in \mathcal{F} \mid n \in N_B, c \in N_I, r \in N_R^+, \pi_1(t) &= \{A \in N_C \mid A(n) \in \mathcal{A}_i\} \cup \{\top\}, \\ \pi_2(t) &= \{r \in N_R^+ \mid r(n,c) \in \mathcal{A}_i\}, \\ \pi_3(t) &= \{A \in N_C \mid A(c) \in \mathcal{A}_i\} \cup \{\top\}\}. \end{aligned}$$

Construct the pre-good successor relation, *$psucc_i \subseteq \mathcal{F} \times \mathcal{P}(\mathcal{S})$, initially set to $(t, \{\}) \in psucc_i$ for all $t \in F_i$, and the* loop computation relation, *$L \subseteq \mathcal{F} \times N_C$, initially empty, for each $i \geq 1$ by applying the following rules exhaustively.*

1. *If $\{(t, A_1), \ldots, (t, A_n)\} \subseteq L$ and $A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B \in \mathcal{T}$, then update L by adding (t, B);*
2. *If $(t, \overline{u}) \in psucc_i$ and $A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B \in \mathcal{T}$ such that there exists $1 \leq j \leq m$, with $\overline{A} \subseteq \pi_2(u_j)$, but $B \notin \pi_2(u_j)$, then update the entry $(t, \overline{u})$ in $psucc_i$ to $(t, \{u_1, \ldots, (\pi_1(u_j), \pi_2(u_j) \cup \{B\}), \ldots, u_m\})$;*
3. *If $(t, \overline{u}) \in psucc_i$ and $\exists r.A \sqsubseteq B \in \mathcal{T}$ such that there exists $1 \leq j \leq m$, with $r \in \pi_1(u_j)$ and $A \in \pi_2(u_j)$, but $B \notin \pi_1(t)$, then update L by adding (t, B);*
4. *If $(t, \overline{u}) \in psucc_i$ and $\exists r.A \sqsubseteq B \in \mathcal{T}$ such that there exists $1 \leq j \leq m$, with $r^- \in \pi_1(u_j)$ and $A \in \pi_1(t) \cup \{A' \mid (t, A') \in L\}$, but $B \notin \pi_2(u_j)$, then update the entry $(t, \overline{u})$ in $psucc_i$ to $(t, \{u_1, \ldots, (\pi_1(u_j), \pi_2(u_j) \cup \{B\}), \ldots, u_m\})$;*
5. *If $(t, \overline{u}) \in psucc_i$ and $r \sqsubseteq r' \in \mathcal{T}$ such that there exists $1 \leq j \leq m$, with $r \in \pi_1(u_j)$, but $r' \notin \pi_1(u_j)$, or $r^- \in \pi_1(u_j)$, but $r'^- \notin \pi_1(u_j)$ then update the entry $(t, \overline{u})$ in $psucc_i$ to $(t, \{u_1, \ldots, (\pi_1(u_j) \cup \{r'\}, \pi_2(u_j)), \ldots, u_m\})$;*
6. *If $\{(t, \overline{u}), (t', \overline{u}')\} \subseteq psucc_i$ such that there exists $1 \leq j \leq m$, with $\pi_1(t) = \pi_3(t')$, $\pi_1(u_j)^- = \pi_2(t')$ and $\pi_2(u_j) = \pi_1(t')$, then update $psucc_i(t)$ to $(t, \{u_1, \ldots, (\pi_1(u_j), \pi_2(u_j) \cup \{A' \mid (t', A') \in L\}), \ldots, u_m\})$;*
7. *If $(t, \overline{u}) \in psucc_i$ and $A \sqsubseteq \exists r.B \in \mathcal{T}$ such that $A \in \pi_1(t)$, and $r \notin \pi_2(t)$ or $A \notin \pi_3(t)$, and for all $1 \leq j \leq m$, $r \notin \pi_1(u_j)$ or $B \notin \pi_2(u_j)$, then update the entry $(t, \overline{u})$ in $psucc_i$ to $(t, \overline{u} \cup \{(\{r\}, \{\top, B\})\})$;*

$$\begin{tikzcd}
 & \pi_3(t) & \\
 & \pi_1(t) \arrow[u, "\pi_2(t)"] \arrow[dl, "\pi_1(u)"'] \arrow[dr, "\pi_1(u')"] & \\
\pi_2(u) & & \pi_2(u')
\end{tikzcd}$$

Figure 1: Structure of the good successor configuration

$$\begin{aligned}
c^{\mathcal{A},S} &= \{c\} \\
(\neg c)^{\mathcal{A},S} &= \Delta^{\mathcal{A}} \setminus \{c\} \\
A^{\mathcal{A},S} &= \{c \mid A(c) \in \mathcal{A}\} \\
(\neg A)^{\mathcal{A},S} &= \Delta^{\mathcal{A}} \setminus \{c \mid A(c) \in \mathcal{A}\} \\
(\varphi_1 \wedge \varphi_2)^{\mathcal{A},S} &= (\varphi_1)^{\mathcal{A},S} \cap (\varphi_2)^{\mathcal{A},S} \\
(\exists E.\varphi)^{\mathcal{A},S} &= \{e \in \Delta^{\mathcal{A}} \mid \exists e' : (e,e') \in E^{\mathcal{A}} \wedge e' \in \varphi^{\mathcal{A},S}\} \\
s^{\mathcal{A},S} &= \{c \mid s(c) \in S\}
\end{aligned}$$

Figure 2: Evaluating shape expressions

8. *If $(t, \overline{u}) \in psucc_i$ and there exists $1 \leq j, k \leq l$ such that $j \neq k$, $\pi_1(u_j) \subseteq \pi_1(u_k)$ and $\pi_2(u_j) \subseteq \pi_2(u_k)$, then update the entry $(t, \overline{u})$ in $psucc_i$ to $(t, \overline{u} \setminus \{u_j\})$;*

9. *If $(t, \overline{u}) \in psucc_i$, then for each $1 \leq j \leq l$, such that $((inv(u_j), \pi_1(t)), \overline{u}') \notin psucc_i$, for any $\overline{u}'$, add $((inv(u_j), \pi_1(t), \{\})$ to $psucc_i$;*

where we use the notation $\overline{A}$, $\overline{u}$, $\overline{u}'$ as a shorthand for $\{A_1, \dots, A_n\}$, $\{u_1, \dots, u_m\}$, resp. $\{u'_1, \dots, u'_{m'}\}$. Now, let $\mathcal{B}_i$ be the smallest set of atoms such that $\mathcal{A}_i \subseteq \mathcal{B}_i$ and, if $(t, B) \in L$ and there exists $c \in N_B$, $d \in N_I$ such that $\pi_1(t) \subseteq \{A \in N_C \mid A(c) \in \mathcal{B}_i\} \cup \{\top\}$, $\pi_2(t) \subseteq \{r \in N_R^+ \mid r(c,d) \in \mathcal{B}_i\}$ and $\pi_3(t) \subseteq \{A \in N_C \mid A(d) \in \mathcal{B}_i\} \cup \{\top\}$, then $B(c) \in \mathcal{B}_i$. Define $\mathcal{A}_{i+1}$ as the unique (up to isomorphism) set of atoms such that $\mathcal{B}_i (\xrightarrow{\mathcal{T}, N_I})^{\omega} \circ \xrightarrow{core} \mathcal{A}_{i+1}$.

Proposition 1. *There exists an $n \geq 1$ such that $\mathcal{A}_n = \mathcal{A}_{n+1}$. We use the notation $\mathcal{A}_{\mathcal{T}}^+$ for $\mathcal{A}_n$.*

First, note that by construction, $\mathcal{A}_i \subseteq \mathcal{A}_{i+1}$. Furthermore, the amount of blank nodes neighbouring a node in the data, such that they do not subsume each other, is restricted by the amount of existential axioms in $\mathcal{T}$.

We can now define the good successor configuration as the limit of $psucc_n$, and the set $\mathcal{F}_{\mathcal{T},\mathcal{A}}$ of *live types* as the types occurring in it.

Definition 3. *Given an $\mathcal{ELHI}$ knowledge base $(\mathcal{T}, \mathcal{A})$. Let $n \geq 1$ be the smallest n such that $\mathcal{A}_n = \mathcal{A}_{n+1}$. Set $succ_{\mathcal{T},\mathcal{A}} = psucc_n$ and let $\mathcal{F}_{\mathcal{T},\mathcal{A}} \subseteq \mathcal{F}$ be such that $t \in \mathcal{F}_{\mathcal{T},\mathcal{A}}$ iff there exists $\overline{u}$ such that $(t, \overline{u}) \in succ_{\mathcal{T},\mathcal{A}}$.*

In the good successor configuration, we only add successors that are directly required by the axioms and whose set of roles can be summarised by one role that implies all others. This will be useful later. We denote by $cl_{\mathcal{T}}(r) \subseteq N_R^+$ the smallest set such that $r \in cl_{\mathcal{T}}(r)$ and if $r' \in cl_{\mathcal{T}}(r)$ and $r' \sqsubseteq r'' \in \mathcal{T}$ or $r'^- \sqsubseteq r''^- \in \mathcal{T}$, then $r'' \in cl_{\mathcal{T}}(r)$.

Proposition 2. *Given an $\mathcal{ELHI}$ knowledge base $(\mathcal{T}, \mathcal{A})$. If $succ_{\mathcal{T},\mathcal{A}}(t, \overline{u})$, then for each $u \in \overline{u}$ there exists a $A \sqsubseteq \exists r.B \in \mathcal{T}$ such that $A \in \pi_1(t)$, $cl_{\mathcal{T}}(r) = \pi_1(u)$ and $B \in \pi_2(u)$.*

When computing the good successor configuration, we are computing all loops, that is, all implications that may propagate back during the chase. In a nutshell, using the good successors is analogous to doing the chase with the closure of $\mathcal{T}$ under semantic consequences $\mathcal{T} \models C \sqsubseteq D$.

Proposition 3. *Let $\mathcal{T}^* := \{C \sqsubseteq D \mid \mathcal{T} \models C \sqsubseteq D\}$, and let $\mathcal{B}$ be the unique (up to isomorphism) set of atoms such that $\mathcal{A}(\xrightarrow{\mathcal{T}^*,N_I})^\omega \circ \xrightarrow{core} \mathcal{B}$. Then $\mathcal{A}^+_{\mathcal{T}} \cong \mathcal{B}$.*

The proof relies on the fact that all relevant loop computations are covered in the construction of the good successor configuration. As we consider live types, all possible configurations in the anonymous parts are considered and all inferred loops are saved in the loop computation list.

4. Core Universal Model Construction

The good successor configuration allows us to easily build the desired universal core model.

We introduce some additional notation. Let $\mathcal{A}$ be a set of atoms. For $X \subseteq N_I \cup N_B$, we define the set $n_{\mathcal{A}}(X)$ of neighbours of X. We also define the type $type_{\mathcal{A}}(c)$ of a blank node $c \in N_B$ with $|n_{\mathcal{A}}(c)| = 1$, where $d \in n_{\mathcal{A}}(c)$.

$$\begin{aligned} n_{\mathcal{A}}(X) &:= \bigcup_{c \in X} \{x \in N_I \cup N_B \mid r(c,x) \in \mathcal{A} \vee r(x,c) \in \mathcal{A}, r \in N_R\} \\ type_{\mathcal{A}}(c) &:= (\{A \in N_C \mid A(c) \in \mathcal{A}\} \cup \{\top\}, \{r \in N_R^\pm \mid r(c,d) \in \mathcal{A}\}, \\ &\qquad\qquad\qquad\qquad\qquad\qquad \{A \in N_C \mid A(d) \in \mathcal{A}\} \cup \{\top\}) \end{aligned}$$

We can now build the model, starting from $\mathcal{A}^+_{\mathcal{T}}$ and adding successors according to $succ_{\mathcal{T},\mathcal{A}}$.

Definition 4. *Given an $\mathcal{ELHI}$ knowledge base $(\mathcal{T}, \mathcal{A})$, define the core universal model $core(\mathcal{T}, \mathcal{A})$ as the union of the sequence $X_1, X_2, \ldots$, that is recursively defined in the following way*

- $X_1 := \mathcal{A}^+_{\mathcal{T}}$;
- *X_{n+1} is constructed from X_n by adding, for each $c \in N_B$ such that $|n_{X_n}(c)| = 1$ and $succ_{\mathcal{T},\mathcal{A}}(type_{X_n}(c), \overline{u})$ holds for some $\overline{u}$, the following atoms for each $u \in \overline{u}$, with a fresh $d \in N_B$ for each $u \in \overline{u}$*
  - *$\{r(c,d), r^-(d,c)\} \subseteq X_{n+1}$ for each $r \in \pi_1(u)$;*
  - *$A(d) \in X_{n+1}$ for each $A \in \pi_2(u)$.*

Note that we can see each entry in the sequence $X_1, X_2, \ldots$ as layers such that the freshly added blank nodes in X_i have distance i to the ABox.

We can now show that $core(\mathcal{T}, \mathcal{A})$ is a model, that it is a core, and that it is universal.

Proposition 4. *For all $\mathcal{ELHI}$ knowledge bases $(\mathcal{T}, \mathcal{A})$, $core(\mathcal{T}, \mathcal{A})$ is a model of $(\mathcal{T}, \mathcal{A})$.*

The proof is based on the idea that all axioms that we consider are all in normal form. Thus, the rules for the pre-good successor relation suffice to ensure that all axioms are satisfied in the part outside of the ABox $\mathcal{A}$. Closer to the data, the result follows from Proposition 3.

Proposition 5. *For all $\mathcal{ELHI}$ knowledge bases $(\mathcal{T}, \mathcal{A})$, $core(\mathcal{T}, \mathcal{A})$ is a core.*

Proof. A set of atoms $\mathcal{A}$ has a *core cover* when there exist finite sets of atoms $\mathcal{B}_1 \subseteq \mathcal{B}_2 \subseteq \ldots$ with $\mathcal{A} = \bigcup_{i \geq 1} \mathcal{B}_i$, such that for all $\mathcal{B}_i$, each homomorphism $h : \Delta^{\mathcal{B}_i} \to \Delta^{\mathcal{A}}$ is an embedding.

From [22], Theorem 16, we learn that each set of atoms that has a core cover is a core. Thus, it suffices to show that $\mathcal{B}_i = X_i$ is a core cover for

$$\mathcal{B} = \bigcup_{i \geq 1} X_i = \mathit{core}(\mathcal{T}, \mathcal{A}).$$

It is immediate that for each i, the identity mapping $h_i : \Delta^{X_i} \to \Delta^{\mathit{core}(\mathcal{T},\mathcal{A})}$ given by $x \mapsto x$ is an embedding. By induction on i it is shown that each homomorphism from X_i to $\mathit{core}(\mathcal{T}, \mathcal{A})$ is equal to the identity mapping.

- $i = 1$. By definition $X_1 = \mathcal{A}^+_{\mathcal{T}}$, which is by construction a core. Note that for all endormorphisms on cores it holds that if one node is mapped to itself, all nodes have to be mapped to themselves. Furthermore, note that each $x \in \Delta^{\mathit{core}(\mathcal{T},\mathcal{A})} \setminus \Delta^{\mathcal{A}^+_{\mathcal{T}}}$ does not have a neighbour in the ABox, so the image of each homomorphism from $\mathcal{A}^+_{\mathcal{T}}$ to $\mathit{core}(\mathcal{T}, \mathcal{A})$ is contained in $\mathcal{A}^+_{\mathcal{T}}$. As all nodes in $\Delta^{\mathcal{A}^+_{\mathcal{T}}} \cap N_I$ have to be mapped to themselves, we can conclude the required.
- $i = n + 1$. Let us consider some $x \in \Delta^{X_{n+1}} \setminus \Delta^{X_n}$. By induction hypothesis, we know that for all homomorphisms $h : \Delta^{X_{n+1}} \to \Delta^{\mathit{core}}$, $h(x') = x'$ for all $x' \in \Delta^{X_n}$. Moreover, we know there exists $y \in \Delta^{X_n}$ with $\mathit{succ}_{\mathcal{T},\mathcal{A}}(\mathit{type}_{X_n}(y), \overline{u})$ such that there exists $u \in \overline{u}$ that led to the introduction of x, i.e., $n_{X_{n+1}}(x) = \{y\}$. So, for each homomorphism $h : \Delta^{X_{n+1}} \to \Delta^{\mathit{core}(\mathcal{T},\mathcal{A})}$, we find $h(x) \in n_{\mathit{core}(\mathcal{T},\mathcal{A})}(y)$. By construction, the neighbourhood of y exists of (i) z such that $n_{X_n}(y) = \{z\}$, however, $h(x) = z$ would mean that the preconditions of introducing a new successor type in rule 7 in Definition 2 were not applied properly, and (ii) x' such that there exists $u' \in \overline{u}$ with $u \neq u'$ that led to the introduction of x', however, $h(x) = x'$ would mean that rule 8 of Definition 2 was not properly applied. Thus, we must conclude that $h(x) = x$ for all $x \in \Delta^{X_{n+1}}$, which concludes this induction. □

Proposition 6. *For all $\mathcal{ELHI}$ knowledge bases $(\mathcal{T}, \mathcal{A})$, $\mathit{core}(\mathcal{T}, \mathcal{A})$ is universal.*

The idea of the proof is that we only introduce new successors if there exists an applicable existential axiom, and that all information in $\mathit{core}(\mathcal{T}, \mathcal{A})$ is a direct consequence of the applicable axioms and will be derived in every model.

Finally, we show that if the core chase terminates, it results in $\mathit{core}(\mathcal{T}, \mathcal{A})$.

Theorem 1. *Given an $\mathcal{ELHI}$ knowledge base $(\mathcal{T}, \mathcal{A})$ such that $\mathcal{B}$ is the unique, up to isomorphism, structure such that $\mathcal{A}(\xrightarrow{\mathcal{T}} \circ \xrightarrow{\mathit{core}})^{\omega}\mathcal{B}$, then $\mathcal{B} \cong \mathit{core}(\mathcal{T}, \mathcal{A})$.*

Proof. First, note that both $\mathcal{B}$, by definition, and $\mathit{core}(\mathcal{T}, \mathcal{A})$, by Proposition 5 are cores. Therefore, as $\mathcal{B}$ is finite, it suffices to show that there exists homomorphisms in both directions. Note that $\mathit{core}(\mathcal{T}, \mathcal{A})$ is a model by Proposition 4. As $\mathcal{B}$ is a universal model, [7], it follows that there exists a homomorphism from $\mathcal{B}$ into each model, including $\mathit{core}(\mathcal{T}, \mathcal{A})$. By Proposition 6 we also have that $\mathit{core}(\mathcal{T}, \mathcal{A})$ is universal, which means also the other required homomorphism must exist. □

The authors of [22] show that the stable chase does not always yield a universal model for existential rules, but it is unclear whether the same result holds for the restriction to $\mathcal{ELHI}$. If the stable chase for $\mathcal{ELHI}$ is universal, then Theorem 1 can be extended to the stable chase.

The core chase construction described in [7] uses multiple *core* computations. This makes it unsuitable for implementation, as computing cores is hard in general [21]. The restricted chase yields a core chase if the axioms are *core-stratified* [23], but there is no efficient algorithm for determining the core-stratification order or even whether the axioms are core-stratified. Note that the few core computations used in Definition 2 are very restricted and are thus not as hard: our approach is one of the first efficient techniques for constructing the core chase. We note that an effective procedure for $\mathcal{ELH}$ was presented in [12], but the authors do not prove that their construction results in the core chase.

5. SHACL Validation

We now consider the problem of SHACL validation in the presence of $\mathcal{ELHI}$ ontologies.

First we introduce the fragment of SHACL we consider, which we call semi-positive SHACL. Like in [15], we follow the formalisation of SHACL in [17]. Let N_S be an infinite set of shape names, disjoint from N_I, N_B, N_C^+ and N_R^+. A *shape expression* φ is built in the following way

$$\varphi ::= c \mid \neg c \mid A \mid \neg A \mid \varphi \wedge \varphi \mid \exists E.\varphi \mid s,$$

for $c \in N_I$, $A \in N_C$, $s \in N_S$ and E a regular path expression such that $E ::= r \mid E \cup E \mid E \circ E \mid E^*$, for $r \in N_R^+$, where "$\cup$" is the standard union of relations, "$\circ$" concatenation as described in the preliminaries and "$*$" the Kleene star. A *constraint* is an expression of the form $s \leftarrow \varphi$, for $s \in N_S$ and φ a shape expression. A *shapes graph* $(\mathcal{C}, \mathcal{G})$ is a pair consisting of a set of constraints $\mathcal{C}$ and a set of *targets* $\mathcal{G}$. Targets are of the form $s(c)$, where $s \in N_S$ and $c \in N_I$. Given a set of atoms $\mathcal{A}$, a *shape assignment* is a set of shape atoms of the form $s(c)$, for $s \in N_S$, $c \in \Delta^{\mathcal{A}}$. Given a shape expressions φ, a shape assignment S, and a set of atoms $\mathcal{A}$, define $\varphi^{\mathcal{A},S} \subseteq \Delta^{\mathcal{A}}$ inductively as in Figure 2, where $(e, e') \in E^{\mathcal{A}}$ if there exists $w_1 \cdot \ldots \cdot w_n$ in the language defined by E and $\{w_1(e, e_1), \ldots, w_n(e_{n-1}, e')\} \subseteq \mathcal{A}$.

Just like in [15], we consider the *least fixed point semantics* for SHACL. We define an immediate consequence operator $T_{\mathcal{A},\mathcal{C}}$ that maps shape assignments to shape assignments: $T_{\mathcal{A},\mathcal{C}}(S) := S \cup \{s(a) \mid s \leftarrow \varphi \in \mathcal{C}, a \in (\varphi)^{\mathcal{A},S}\}$. For a set of atoms $\mathcal{A}$ and a shapes graph $(\mathcal{C}, \mathcal{G})$, we say that $\mathcal{A}$ validates $(\mathcal{C}, \mathcal{G})$ when $\mathcal{G}$ is contained in the least fixed point of $T_{\mathcal{A},\mathcal{C}}$. In the presence of a TBox $\mathcal{T}$, we define the semantics of validation using the core universal model. That is, we say that $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$ in case $core(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$.

Our goal is now to reduce validation in the presence of a TBox, to plain validation over a set of atoms. The following normal form for SHACL constraints will facilitate this goal.

Definition 5. *A SHACL constraint is in normal form if it has one of the following forms*

$$\begin{array}{lll} \textit{(NC1)}\ s \leftarrow c & \textit{(NC2)}\ s \leftarrow \neg c & \textit{(NC3)}\ s \leftarrow A \\ \textit{(NC4)}\ s \leftarrow \neg A & \textit{(NC5)}\ s \leftarrow s' \wedge s'' & \textit{(NC6)}\ s \leftarrow \exists r.s', \end{array}$$

where $c \in N_I$, $r \in N_R^+$, $\{s, s', s''\} \subseteq N_S$ and $A \in N_C$.

Proposition 7. *Each set of semi-positive SHACL constraints $\mathcal{C}$ can be translated in time polynomial in $|\mathcal{C}|$, into a set of semi-positive constraints $\mathcal{C}'$ in normal form such that for all $\mathcal{G}$ that do not use the freshly introduced shape names, and all $\mathcal{T}$ and $\mathcal{A}$, we have that $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$ iff $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}', \mathcal{G})$.*

Proof. With a similar construction as of Proposition 4.2 in [18], it is easy to see that each constraint can be translated in polynomial time into the a normal form consisting of (NC1) until (NC5) as above, and (NC6') $s \leftarrow \exists E.s'$. To normalise $s \leftarrow \exists E.s'$, suppose $\mathcal{M} = (Q, \Sigma, q_I, \Delta, q_F)$ is the nondeterministic finite automaton recognising E. Take fresh shape names s_q for each $q \in Q$ and add the following constraints $s \leftarrow s_{q_I} \wedge s_{q_I}$, $s_q \leftarrow \exists r.s'_q$ if $(q, r, q') \in \Delta$ and $s_{q_F} \leftarrow s' \wedge s'$. Note that this automaton can be constructed in polynomial time, which is a standard result in the literature, see for instance [24]. □

We are ready now to reduce validation to the case without a TBox. Given a TBox $\mathcal{T}$ and a set of atoms $\mathcal{A}$, or more specifically, the $succ_{\mathcal{T},\mathcal{A}}$ that represents a universal core model of $(\mathcal{T}, \mathcal{A})$. We transform a shapes graph $(\mathcal{C}, \mathcal{G})$ into a new shapes graph $(\mathcal{C}_{\mathcal{T},\mathcal{A}}, \mathcal{G})$ such that $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$ iff $\mathcal{A}'$ validates $(\mathcal{C}_{\mathcal{T},\mathcal{A}}, \mathcal{G})$, for some set of atoms $\mathcal{A}'$ that can be easily obtained from $\mathcal{A}$ and the construction of $succ_{\mathcal{T},\mathcal{A}}$. Unlike the similar rewriting for *DL-Lite*$_{\mathcal{R}}$, the variant we introduce here does not work for *every* set of atoms $\mathcal{A}$: we can only guarantee soundness and completeness if the set of atoms $\mathcal{A}$ is such that $succ_{\mathcal{T},\mathcal{A}}$ represents the core universal model of $(\mathcal{T}, \mathcal{A})$. We compromise the data independence, but in this way, we avoid the best case exponentially of its predecessor [15].

We use a fresh concept name $\widehat{A} \in N_C$ to decorate the outer layer of $\mathcal{A}^{+}_{\mathcal{T}}$, and define $\mathcal{A}^{*}_{\mathcal{T}} := \mathcal{A}^{+}_{\mathcal{T}} \cup \{\widehat{A}(c) \mid c \in \Delta^{\mathcal{A}^{+}_{\mathcal{T}}} \cap N_B\}$.The elements in our rewriting are triples $(t, W, H) \subseteq \mathcal{F}_{\mathcal{T},\mathcal{A}} \times \mathcal{P}(\{\exists r.s \mid r \in N^{+}_{R}, s \in N_S\}) \times \mathcal{P}(N_S)$: t is a live type, W a set of assumptions and H a set of shape names. The starting set I of our rewriting is such that $(t, W, H) \in I$ iff

- if $\exists r.s' \in W$, there exists $s \leftarrow \exists r.s' \in \mathcal{C}$ and $r \in \pi_2(t)$, and
- $s \in H$ iff there exists $\exists r.s' \in W$ such that $s \leftarrow \exists r.s' \in \mathcal{C}$.

Our rewriting algorithm adds shape names s to the heads H. Specifically, we update $(t, W, H) \in I$, to $(t, W, H \cup \{s\})$ when any of the following holds:

1. $s \leftarrow A \in \mathcal{C}$, $A \in \pi_1(t)$, $s \notin H$, or
2. $s \leftarrow \neg A \in \mathcal{C}$, $A \notin \pi_1(t)$, $s \notin H$, or
3. $s \leftarrow \neg c \in \mathcal{C}$, $s \notin H$, or
4. $s \leftarrow s_1 \wedge s_2 \in \mathcal{C}$, $\{s_1, s_2\} \subseteq H$, $s \notin H$, or
5. $s \leftarrow \exists r.s_1 \in \mathcal{C}$, and there exists $(t', W', H') \in I$ such that, for some $u \in \overline{u}$ with $succ_{\mathcal{T},\mathcal{A}}(t, \overline{u})$, we have $t' = (inv(u), \pi_1(t))$, $s_1 \in H'$, $r^- \in \pi_2(t)$ and $\{s \mid \exists r.s \in W'\} \subseteq H$.

Let $sat(I)$ be the result of exhaustively adding shape names as above. The triples in $sat(I)$ are then translated into the set of constraints $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ in the following way.

If $(t, W, H) \in sat(I)$, then for each $s \in H$ the following constraint is contained in $\mathcal{C}_{\mathcal{T},\mathcal{A}}$

$$s \leftarrow \bigwedge_{A \in \pi_1(t)} A \wedge \exists r. \bigwedge_{B \in \pi_3(t)} B \wedge \bigwedge_{\exists r.s' \in W} \exists r.s' \wedge \widehat{A},$$

where $r \in \pi_2(t)$ is such that for all $r' \in \pi_2(t)$ we have $r' \in cl_{\mathcal{T}}(r)$. Note that the existence of r is guaranteed by Proposition 2.

This gives us the desired rewriting. Its proof of correctness is analogous to the one in [15].

Theorem 2. *Given an $\mathcal{ELHI}$ knowledge base $(\mathcal{T}, \mathcal{A})$ and $(\mathcal{C}, \mathcal{G})$ a semi-positive SHACL shapes graph. Then it holds that $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$ iff $\mathcal{A}^*_\mathcal{T}$ validates $(\mathcal{C}_{\mathcal{T},\mathcal{A}}, \mathcal{G})$.*

Since we are taking the input $\mathcal{A}$ into account, we do not have to consider all possible types and $\mathcal{C}_\mathcal{T}$ is not best-case exponential. However, there are still ways to optimise the rewriting. The first and simplest optimisation is to restrict the form of the starting sets even more, that is, not all $s \in N_S$ will ever interact with each other. So, let $rel(s) \subseteq N_S$ be defined as the smallest set such that $s \in rel(s)$; if $s' \leftarrow s'' \wedge s''' \in \mathcal{C}$ and $\{s', s'', s'''\} \cap rel(s) \neq \{\}$, then $\{s', s'', s'''\} \in rel(s)$; and if $s' \leftarrow \exists r.s'' \in \mathcal{C}$ and $\{s', s''\} \cap rel(s) \neq \{\}$, then $\{s', s''\} \in rel(s)$. The restriction is then that (t, W, H) is only considered to be in the starting set in case there exists $s \in N_S$ such that $H \subseteq rel(s)$. This optimisation is mostly useful for sets of constraints that relatively use a small amount of role names. Another more involved optimisation is based on the close correspondence between the SHACL fragment we are considering and $\mathcal{ELHI}$ in normalised form. We can translate the SHACL constraints into $\mathcal{ELHI}$ axioms, introducing for each existing concept name a fresh concept name to denote the absence of this concept name in $succ_{\mathcal{T},\mathcal{A}}$. A similar algorithm to the one in Section 3 will then decorate the good successor configuration with all relevant shape names. We believe that the latter version may be better suited for implementation.

However, all these optimisations cannot avoid ExpTime-complexity in the worst case. The proof of the next claim is an easy generalisation of the one in [15]. To show PTime-completeness in data complexity the rewriting can be made data independent as in [15].

Theorem 3. *In the presence of $\mathcal{ELHI}$ TBoxes, SHACL validation is ExpTime-complete in combined complexity and PTime-complete in data complexity. This holds also for positive constraints.*

6. Conclusions and Outlook

In this paper, we introduced a new, more efficient representation of a core universal model. We had shown in [15] that this can be done in a data-independent way; the construction there is for *DL-Lite*$_\mathcal{R}$, but a data-independent computation for $\mathcal{ELHI}$ would not be harder than the one we have presented here. Instead, by considering the exact configurations that appear in the data, we have chosen to focus solely on computing the relevant parts, highly reducing the number of good successor configurations and TBox subsumptions that need to be computed. Thus, this is a first step towards an efficient implementation of the core chase.

We further focus on one of many application areas of this technique: we use the alternative representation of a core universal model to reduce SHACL validation combined with $\mathcal{ELHI}$ TBoxes to plain SHACL validation. Also here, we could take all types for a data-independent rewriting, but restricting it to the live types allows us to avoid an exponential computation in many cases, as not all combinations of types and assumptions need to be considered. This means the proposed algorithm also brings us closer to implementing the combination of SHACL with ontologies. We are confident that the proposed techniques can most likely be optimised further, but leave this for future work. To keep the presentation simple we focused on semi-positive SHACL in this paper, but we note that the rewriting presented in the last section can be extended to SHACL with stratified negation following the ideas of [15].

Acknowledgments

The project leading to this application has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101034440.

This work was also partially supported by the Austrian Science Fund (FWF) project P30873.

References

[1] M. Ortiz, M. Simkus, Reasoning and query answering in description logics, in: Reasoning Web, volume 7487 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 1–53.

[2] M. Bienvenu, M. Ortiz, Ontology-mediated query answering with data-tractable description logics, in: Reasoning Web, volume 9203 of *Lecture Notes in Computer Science*, Springer, 2015, pp. 218–307.

[3] M. Ortiz, S. Rudolph, M. Simkus, Query answering in the horn fragments of the description logics SHOIQ and SROIQ, in: IJCAI, IJCAI/AAAI, 2011, pp. 1039–1044.

[4] D. S. Johnson, A. C. Klug, Testing containment of conjunctive queries under functional and inclusion dependencies, J. Comput. Syst. Sci. 28 (1984) 167–189.

[5] R. Fagin, P. G. Kolaitis, R. J. Miller, L. Popa, Data exchange: semantics and query answering, Theor. Comput. Sci. 336 (2005) 89–124.

[6] B. Marnette, Generalized schema-mappings: from termination to tractability, in: J. Paredaens, J. Su (Eds.), Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009, June 19 - July 1, 2009, Providence, Rhode Island, USA, ACM, 2009, pp. 13–22.

[7] A. Deutsch, A. Nash, J. B. Remmel, The chase revisited, in: M. Lenzerini, D. Lembo (Eds.), Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2008, June 9-11, 2008, Vancouver, BC, Canada, ACM, 2008, pp. 149–158.

[8] T. Eiter, M. Ortiz, M. Simkus, T. Tran, G. Xiao, Query rewriting for Horn-SHIQ plus rules, in: J. Hoffmann, B. Selman (Eds.), Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada, AAAI Press, 2012, pp. 726–733.

[9] H. Pérez-Urbina, B. Motik, I. Horrocks, Tractable query answering and rewriting under description logic constraints, J. Appl. Log. 8 (2010) 186–209.

[10] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi, D. F. Savo, The MASTRO system for ontology-based data access, Semantic Web 2 (2011) 43–53.

[11] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, G. Xiao, Ontop: Answering SPARQL queries over relational databases, Semantic Web 8 (2017) 471–487.

[12] S. Borgwardt, W. Forkel, Closed-world semantics for conjunctive queries with negation over ELH-bottom ontologies, in: S. Kraus (Ed.), Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019, ijcai.org, 2019, pp. 6131–6135.

[13] M. Arenas, G. Gottlob, A. Pieris, Expressive languages for querying the semantic web, ACM Trans. Database Syst. 43 (2018) 13:1–13:45.

[14] S. Ellmauthaler, M. Krötzsch, S. Mennicke, Answering queries with negation over existential rules, in: Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, AAAI Press, 2022, pp. 5626–5633.

[15] S. Ahmetaj, M. Ortiz, A. Oudshoorn, M. Šimkus, Reconciling shacl and ontologies: Semantics and validation via rewriting, in: ECAI 2023, IOS Press, 2023, pp. 27–35.

[16] W3C, Shape constraint language (SHACL), Technical Report. (2017). https://www.w3.org/TR/shacl.

[17] J. Corman, J. L. Reutter, O. Savkovic, Semantics and validation of recursive SHACL, in: D. Vrandecic, K. Bontcheva, M. C. Suárez-Figueroa, V. Presutti, I. Celino, M. Sabou, L. Kaffee, E. Simperl (Eds.), The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I, volume 11136 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 318–336.

[18] M. Andresel, J. Corman, M. Ortiz, J. L. Reutter, O. Savkovic, M. Šimkus, Stable model semantics for recursive SHACL, in: Proc. of The Web Conference 2020, ACM, 2020, p. 1570–1580.

[19] P. Pareti, G. Konstantinidis, F. Mogavero, Satisfiability and containment of recursive SHACL, Journal of Web Semantics 74 (2022) 100721.

[20] B. Bogaerts, M. Jakubowski, J. Van den Bussche, SHACL: A description logic in disguise, in: Proc. of LPNMR 2022, volume 13416 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 75–88.

[21] P. Hell, J. Nesetril, The core of a graph, Discret. Math. 109 (1992) 117–126.

[22] D. Carral, M. Krötzsch, M. Marx, A. Ozaki, S. Rudolph, Preserving constraints with the stable chase, in: B. Kimelfeld, Y. Amsterdamer (Eds.), 21st International Conference on Database Theory, ICDT 2018, March 26-29, 2018, Vienna, Austria, volume 98 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, pp. 12:1–12:19.

[23] M. Krötzsch, Computing cores for existential rules with the standard chase and ASP, in: D. Calvanese, E. Erdem, M. Thielscher (Eds.), Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12-18, 2020, 2020, pp. 603–613.

[24] J. Hromkovic, S. Seibert, T. Wilke, Translating regular expressions into small ϵ-free nondeterministic finite automata, J. Comput. Syst. Sci. 62 (2001) 565–588.