Efficient Inferencing for the Description Logic Underlying OWL EL*

Markus Krötzsch

Technical Report 3005
Institute AIFB, Karlsruhe Institute of Technology, DE
mak@aifb.uni-karlsruhe.de

Abstract. The recent OWL 2 W3C recommendation includes the lightweight ontology language OWL EL which is semantically based on an extension of the $\mathcal{EL}^{++}$ description logic (DL). It is widely assumed that inferencing in OWL EL is possible in polynomial time, but it is not obvious how to extend existing reasoning procedures for $\mathcal{EL}^{++}$ accordingly. We set out to close this gap by developing inferencing methods for $\mathcal{SROEL}(\sqcap, \times)$ – a DL that subsumes the main features of OWL EL. We present a framework for studying materialisation calculi based on datalog, and we use it to investigate the resource requirements for inferencing. We can show that certain $\mathcal{SROEL}(\sqcap, \times)$ feature combinations must lead to increased space upper bounds in any materialisation calculus, suggesting that efficient implementations are easier to obtain for suitably chosen fragments of $\mathcal{SROEL}(\sqcap, \times)$.

1 Introduction

The recent OWL 2 W3C recommendation includes the lightweight ontology language OWL EL [11] which is semantically based on an extension of the $\mathcal{EL}^{++}$ description logic (DL). It is widely assumed that inferencing in OWL EL is possible in polynomial time, but it is not obvious how to extend existing reasoning procedures for $\mathcal{EL}^{++}$ accordingly [2]. In this paper, we set out to close this gap by developing suitable inferencing calculi for the DL $\mathcal{SROEL}(\sqcap, \times)$ which can be considered as an extension of the tractable DL $\mathcal{EL}^{++}$ with local reflexivity (Self), conjunctions of roles, and concept products. The latter two features generalise role disjointness, the universal (top) role, and admissible range restrictions as introduced in OWL EL. Concrete domains (datatypes) hardly interact with the additional features of $\mathcal{SROEL}(\sqcap, \times)$ and are not considered in this paper, though the according mechanisms used in [2] could be lifted to $\mathcal{SROEL}(\sqcap, \times)$.

Our second main contribution is to assess the *efficiency* of the proposed calculi. Inferencing for $\mathcal{EL}$-type DLs often suggests a materialisation-based (or consequence-driven) implementation, where all deductions are computed simultaneously in a bottom-up fashion. The number of inferable facts is an important measure of efficiency in this

* The results established herein have been published in: Markus Krötzsch. Efficient Inferencing for OWL EL. In Tomi Janhunen, Ilkka Niemelä (eds.): Proceedings of the 12th European Confernce on Logics in Artificial Intelligence (JELIA'10), LNAI. Springer, 2010. To appear.

Table 1. Syntax and semantics of $\mathcal{SROEL}(\sqcap,\times)$ concept expressions and axioms for an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$

Concept constructor	Syntax	Semantics
top	$\top$	$\Delta^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid y \in \Delta^{\mathcal{I}} \colon \langle x,y\rangle \in R^{\mathcal{I}}, y \in C^{\mathcal{I}}\}$
local reflexivity	$\exists S.\mathsf{Self}$	$\{x \in \Delta^{\mathcal{I}} \mid \langle x,x\rangle \in S^{\mathcal{I}}\}$
nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$
Axiom	**Syntax**	**Semantics**
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertion	$R(a,b)$	$\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$
concept inclusion (GCI)	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
role inclusion	$R \sqsubseteq T$	$R^{\mathcal{I}} \subseteq T^{\mathcal{I}}$
generalised role inclusion	$R \circ S \sqsubseteq T$	$\{\langle x,z\rangle \mid \langle x,y\rangle \in R^{\mathcal{I}}, \langle y,z\rangle \in S^{\mathcal{I}} \text{ for some } y\} \subseteq T^{\mathcal{I}}$
role conjunction	$S_1 \sqcap S_2 \sqsubseteq T$	$S_1^{\mathcal{I}} \cap S_2^{\mathcal{I}} \subseteq T^{\mathcal{I}}$
concept product	$C \times D \sqsubseteq T$	$C^{\mathcal{I}} \times D^{\mathcal{I}} \subseteq T^{\mathcal{I}}$
	$R \sqsubseteq C \times D$	$T^{\mathcal{I}} \subseteq C^{\mathcal{I}} \times D^{\mathcal{I}}$
$C, D \in \mathbf{C}$, $R, S_{(i)}, T \in \mathbf{N_R}$, $a, b \in \mathbf{N_I}$		

case, and we present a formalisation of materialisation calculi to relate it to the space complexity of datalog reasoning. Since upper space bounds for datalog are exponential in the *arity* of inferred predicates, our goal is to find materialisation calculi where these arities are low. We are able to show that there are limits to such optimisation: some inferencing tasks intrinsically require predicates of higher arities than others.

We present four inferencing calculi: a materialisation calculus for instance checking in $\mathcal{SROEL}(\sqcap,\times)$ in Section 3, and three calculi for classification in $\mathcal{SROEL}(\sqcap,\times)$ and two of its fragments in Section 4. Thereafter, in Section 5, we show that the arity of inferred predicates is minimal for each of the presented calculi.

2 Preliminaries

This section summarises the basic notions from DL and datalog that are used in this paper. The main DL studied herein is $\mathcal{SROEL}(\sqcap,\times)$ which subsumes all semantic features of OWL EL that are not related to datatypes (concrete domains). Readers without basic acquaintance to description logics are advised to refer to the literature [4]. Details about relationship of OWL 2 to DLs are found in [7].

A signature of $\mathcal{SROEL}(\sqcap,\times)$ consists of three disjoint finite sets of *individual names* $\mathbf{N_I}$, *concept names* $\mathbf{N_C}$, and *role names* $\mathbf{N_R}$. Given such a signature, the set of $\mathcal{SROEL}(\sqcap,\times)$ *concept expressions* $\mathbf{C}$ is defined inductively to contain the expressions in Table 1 (top). The set of $\mathcal{SROEL}(\sqcap,\times)$ *axioms* is then defined as in Table 1 (bottom). One may distinguish between axioms of *ABox* (assertional axioms), *TBox* (terminological axioms: GCIs), and *RBox* (axioms related to roles).

Knowledge bases are sets of axioms that satisfy some additional properties. Consider a set KB of $\mathcal{SROEL}(\sqcap,\times)$ axioms. We inductively define the set of *non-simple*

roles of KB to contain all roles T for which there is an axiom $R \circ S \sqsubseteq T \in$ KB, or an axiom $R \sqsubseteq T$ such that R is non-simple. A role that is not non-simple is called *simple*. Moreover, given a role name R, we define $\mathsf{ran}(R)$ to denote the set of concept expressions $D \in \mathbf{C}$ for which KB contains axioms $R \sqsubseteq S_1, \ldots, S_{n-1} \sqsubseteq S_n$ and $S_n \sqsubseteq C \times D$ for some $S_1, \ldots, S_n \in \mathbf{N_R}$ and $n \geq 0$. The set KB is a $\mathcal{SROEL}(\sqcap, \times)$ *knowledge base* if the following restrictions are satisfied:

- all roles S occurring in expressions $\exists S.\mathsf{Self} \in$ KB are simple,
- all roles S_1, S_2 occurring in axioms $S_1 \sqcap S_2 \sqsubseteq T \in$ KB are simple,
- for every axiom $R \circ S \sqsubseteq T \in$ KB we have $\mathsf{ran}(T) \subseteq \mathsf{ran}(S)$, and
- for every axiom $S_1 \sqcap S_2 \sqsubseteq T \in$ KB we have $\mathsf{ran}(T) \subseteq \mathsf{ran}(S_1) \cup \mathsf{ran}(S_2)$.

Note that we do not impose the structural restrictions of RBox regularity here [8] which also apply to OWL DL (and hence to OWL EL) ontologies, since these are not needed for efficient reasoning in $\mathcal{SROEL}(\sqcap, \times)$.

The semantics of $\mathcal{SROEL}(\sqcap, \times)$ is specified by defining interpretations $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ where $\Delta^{\mathcal{I}}$ is a non-empty set, and $\cdot^{\mathcal{I}}$ is an interpretation function that maps individuals to elements of $\Delta^{\mathcal{I}}$, concept names to subsets of $\Delta^{\mathcal{I}}$, and role names to binary relations over $\Delta^{\mathcal{I}}$. Interpretations are extended to concept expressions as in Table 1 (top). A $\mathcal{SROEL}(\sqcap, \times)$ axiom is *satisfied* by an interpretation $\mathcal{I}$ if the according condition of Table 1 (bottom) holds. $\mathcal{I}$ satisfies a knowledge base KB, written $\mathcal{I} \models$ KB, if it satisfies all of its axioms. The models of an axiom or knowledge base are the interpretations that satisfy it, and a semantic entailment relation $\models$ is defined as usual.

Concept products on the left-hand side allow us to define the universal (top) role U with an axiom $\top \times \top \sqsubseteq U$. Note that we can also define the empty (bottom) role N using $\exists N.\top \sqsubseteq \bot$. Using the empty role, conjunctions of (simple) roles are a generalisation of disjointness of (simple) roles: the axiom $R \sqcap S \sqsubseteq N$ declares S and R to be disjoint. In the absence of other role conjunctions, our requirements on concept products in $\mathcal{SROEL}(\sqcap, \times)$ knowledge bases agree with the known admissibility requirements for range restrictions in $\mathcal{EL}^{++}$ [3].

To simplify our investigations, we first observe that any $\mathcal{SROEL}(\sqcap, \times)$ knowledge base can be converted into a normal form in a similar fashion as other $\mathcal{EL}$-type logics [2].

Definition 1. *A $\mathcal{SROEL}(\sqcap, \times)$ knowledge base* KB *is in* normal form *if it contains only axioms of one of the following forms:*

$$C(a) \quad R(a,b) \quad A \sqsubseteq \bot \quad \top \sqsubseteq C \quad A \sqsubseteq \{c\} \quad \{a\} \sqsubseteq \{c\}$$
$$A \sqsubseteq C \quad A \sqcap B \sqsubseteq C \quad \exists R.A \sqsubseteq C \quad A \sqsubseteq \exists R.B \quad \exists R.\mathsf{Self} \sqsubseteq C \quad A \sqsubseteq \exists R.\mathsf{Self}$$
$$R \sqsubseteq T \quad R \circ S \sqsubseteq T \quad R \sqcap S \sqsubseteq T \quad A \times B \sqsubseteq R \quad R \sqsubseteq C \times D$$

where $A, B, C, D \in \mathbf{N_C}$, $R, S, T \in \mathbf{N_R}$, and $a, b, c \in \mathbf{N_I}$.

It is well-known that DL knowledge bases can often be transformed into such normal forms in such a way that satisfiability is preserved. Here, we observe that one actually obtains a stronger correspondence that is closely related to the notion of conservative extension:

$\hat{C} \times D$	$\sqsubseteq$	R	$\mapsto$	$\{\hat{C} \sqsubseteq X, X \times D \sqsubseteq R\}$
$C \times \hat{D}$	$\sqsubseteq$	R	$\mapsto$	$\{\hat{D} \sqsubseteq X, C \times X \sqsubseteq R\}$
R	$\sqsubseteq$	$\hat{C} \times D$	$\mapsto$	$\{X \sqsubseteq \hat{C}, R \sqsubseteq X \times D\}$
R	$\sqsubseteq$	$C \times \hat{D}$	$\mapsto$	$\{X \sqsubseteq \hat{D}, R \sqsubseteq C \times X\}$
$\hat{C}$	$\sqsubseteq$	$\hat{D}$	$\mapsto$	$\{\hat{C} \sqsubseteq X, X \sqsubseteq \hat{D}\}$
C	$\sqsubseteq$	$\top$	$\mapsto$	$\emptyset$
$\bot$	$\sqsubseteq$	C	$\mapsto$	$\emptyset$
$\hat{C} \sqcap A$	$\sqsubseteq$	B	$\mapsto$	$\{\hat{C} \sqsubseteq X, X \sqcap A \sqsubseteq B\}$
A	$\sqsubseteq$	$C \sqcap D$	$\mapsto$	$\{A \sqsubseteq C, A \sqsubseteq D\}$
$\exists R.\hat{C}$	$\sqsubseteq$	A	$\mapsto$	$\{\hat{C} \sqsubseteq X, \exists R.X \sqsubseteq A\}$
A	$\sqsubseteq$	$\exists R.\hat{C}$	$\mapsto$	$\{A \sqsubseteq \exists R.X, X \sqsubseteq \hat{C}\}$
$\hat{C}(a)$			$\mapsto$	$\{X(a), X \sqsubseteq \hat{C}\}$
$A, B \in \mathbf{N_C}$, $X \in \mathbf{N_C}$ a fresh concept name, $C, D, \hat{C}, \hat{D} \in \mathbf{C}$ with $\hat{C}, \hat{D} \notin \mathbf{N_C}$, $R \in \mathbf{N_R}$				

Fig. 1. Normal form transformation for $\mathcal{SROEL}(\sqcap, \times)$

Proposition 1. *For every $\mathcal{SROEL}(\sqcap, \times)$ knowledge base* KB*, a $\mathcal{SROEL}(\sqcap, \times)$ knowledge base* KB′ *over an extended signature can be computed in linear time such that all axioms in* KB′ *are in normal form, and, for all $\mathcal{SROEL}(\sqcap, \times)$ axioms α that only use signature symbols from* KB*, we find that* $\mathrm{KB} \models \alpha$ *iff* $\mathrm{KB}' \models \alpha$.

Proof. A transformation is only required for axioms with complex concept expressions, since we already require most RBox axioms to be in normal form by definition.[1] The transformation for axioms that are not in normal form yet is accomplished by exhaustively applying the rules of Fig. 1, where each rule describes the replacement of the axiom on the right-hand side by the set of axioms on the left-hand side. With a *fresh* concept name, we mean one that does not occur in any axiom yet. It is easy to see that only a linear number of transformation steps are required, where it is important to note that the rule for $A \sqsubseteq C \sqcap D$ is only applicable if A is no compound expression, so that the duplication of A still leads to only a linear increase in size.

It is easy to see that, for each transformation rule, the resulting set of axioms has the required semantic relation to the original axiom: Any interpretation that satisfies the original axiom can be extended to an interpretation of the transformed axiom set by interpreting each fresh concept name X as the least set of domain elements for which the transformed set of axioms is valid (using the original interpretation interpretation for all other symbols). Conversely, any interpretation that satisfies the transformed set of axioms necessarily satisfies the original axiom. Since these observations are easily verified for each transformation rule, the claim follows by induction. □

Roughly speaking, the previous result states that every $\mathcal{SROEL}(\sqcap, \times)$ knowledge base is semantically equivalent "up to the interpretation of auxiliary signature symbols" to a knowledge base in normal form.

[1] This could be relaxed by allowing arbitrarily long role chains which can easily be decomposed into binary role chains as in our definition.

Our formalisation of inferencing calculi is based on the simple rule language *datalog* [1]. A *signature* of datalog is a tuple $\langle \mathbf{C}, \mathbf{P} \rangle$, where $\mathbf{C}$ is a finite set of *constants*, and $\mathbf{P}$ is a finite set of *predicates*, where each predicate $p \in \mathbf{P}$ has a fixed arity $\mathsf{ar}(p) \geq 0$. We assume $\mathbf{P}$ to be a disjoint union $\mathbf{P}_i \cup \mathbf{P}_e$ of *IDB predicates* $\mathbf{P}_i$ and *EDB predicates* $\mathbf{P}_e$.[2] Throughout this paper, we use $\mathbf{V}$ to denote a countably infinite set of *variables*. Elements of $\mathbf{C} \cup \mathbf{V}$ are called *terms*.

A *datalog atom* over a signature $\langle \mathbf{C}, \mathbf{P} \rangle$ is an expression $p(t_1, \ldots, t_n)$ where $p \in \mathbf{P}$ with $\mathsf{ar}(p) = n$, and $t_i \in \mathbf{C} \cup \mathbf{V}$ for $i = 1, \ldots, n$. An IDB (EDB) atom is one that uses an IDB (EDB) predicate. A *datalog rule* is a formula of the form $B_1 \wedge \ldots \wedge B_l \rightarrow H$ where B_i and H are datalog atoms, and H is an IDB atom. The premise of a rule is also called its *body*, and the conclusion is called its *head*. A *datalog program* P is a set of datalog rules. A *fact* is a ground, i.e. variable-free, rule with an empty body.

A *ground substitution* σ for a signature $\langle \mathbf{C}, \mathbf{P} \rangle$ is a function $\sigma : \mathbf{V} \rightarrow \mathbf{C}$. Substitutions are extended to datalog atoms by setting $\sigma(p(t_1, \ldots, t_n)) \coloneqq p(\sigma(t_1), \ldots, \sigma(t_n))$, and $\sigma(p(t_1, \ldots, t_n))$ is called a *ground instance* of $p(t_1, \ldots, t_n)$ in this case.

A *proof tree* for a datalog program P is a structure $\langle N, E, \lambda \rangle$ where N is a finite set of nodes, $E \subseteq N \times N$ is a set of edges of a directed tree, and λ is a labelling function that assigns a ground datalog atom to each node, where the following holds: for each node $n \in N$, there is a rule $B_1 \wedge \ldots \wedge B_l \rightarrow H \in P$ and a ground substitution σ such that $\lambda(n) = \sigma(H)$ and the set of child nodes $\{m \mid \langle n, m \rangle \in E\}$ is of the form $\{m_1, \ldots, m_l\}$ where $\lambda(m_i) = \sigma(B_i)$ for each $i = 1, \ldots, l$.

A ground atom H is a *consequence* of a datalog program P if there is a proof tree for P that has H as the label $\lambda(r)$ of its root node r.

Definition 2. *Given a datalog signature* $\langle \mathbf{C}, \mathbf{P} \rangle$*, a* renaming ρ *is a function* $\rho : \mathbf{C} \rightarrow \mathbf{C}$*. To extend* ρ *to ground datalog atoms we set* $\rho(p(t_1, \ldots, t_n)) \coloneqq p(\rho(t_1), \ldots, \rho(t_n))$*.*

All renamings that occur in this paper are injective.

3 Instance Checking for $\mathcal{SROEL}(\sqcap, \times)$

We now introduce a calculus for solving the inference task of instance checking – deciding if $C(a)$ is entailed for any $C \in \mathbf{N_C}$, $a \in \mathbf{N_I}$ – for $\mathcal{SROEL}(\sqcap, \times)$. In Section 5 we show its optimality in the sense that no other materialisation calculus can be better in terms of certain characteristics. This requires a concrete understanding of what a materialisation calculus is, so we start this section with a formalisation of this notion.

Our goal is to find a unified presentation for deduction calculi that have been proposed for $\mathcal{EL}$-type DLs before [2,5]. Intuitively speaking, a materialisation calculus is a system of deduction rules for deriving logical consequences which – as opposed to a complete inferencing algorithm – does not specify a concrete control flow or processing strategy for evaluating these rules. Deduction rules can be denoted in many forms, e.g. using textual if-then descriptions [2], in tabular form [11,6], or as sequent calculus style derivation rules [5]. Premises and conclusions of rules often consist of formulae

[2] This terminology originates from the field of deductive databases where one distinguishes *extensional* and *intensional data base*.

of the processed logic, but they may also contain auxiliary expressions that are relevant to the calculus.[3] A deduction rule can then be viewed as a schema for deriving new expressions from a finite set of given expressions, and its applicability is not affected by uniform renamings of signature symbols in premise and conclusion.

Deduction rules in this sense can conveniently be denoted as datalog rules where concrete logical sentences are represented as ground facts that use signature symbols in term positions. For example, we can represent $A \sqsubseteq B$ as $\texttt{subclassOf}(A, B)$, and introduce a deduction rule $\texttt{subclassOf}(x, y) \wedge \texttt{subclassOf}(y, z) \rightarrow \texttt{subclassOf}(x, z)$. This unifies the presentation of diverse calculi, and allows us to exploit techniques from deductive databases. For connecting datalog to DL, we require a translation from individual DL axioms to (sets of) datalog EDB facts. This translation is also defined for signature symbols, since symbols must generally be "loaded" into datalog to be able to derive conclusions about them, regardless of whether the symbols occurred in input axioms or not. Finally, another translation is used for finding the IDB fact that signifies the logical entailment of a given axiom. All translation functions can be partial if not all types of axioms are supported by the calculus. These considerations motivate the following definition.

Definition 3. *A* materialisation calculus *K is a tuple $K = \langle I, P, O\rangle$ where I and O are partial functions, and P is a set of datalog rules, such that*

1. *given an axiom or signature symbol α, $I(\alpha)$ is either undefined or a set of datalog facts over EDB predicates,*
2. *given an axiom α, $O(\alpha)$ is either undefined or a single datalog fact over an IDB predicate,*
3. *the set of EDB and IDB predicates used by I, P, and O is fixed and finite,*
4. *P contains no constant symbols,*
5. *all constant symbols used in $I(\alpha)$ or $O(\alpha)$ for some axiom (or signature symbol) α are either signature symbols that appear in (or are equal to) α, or constants of the form aux_i^α with $i \geq 0$, where all constant names aux_i^α are mutually distinct and unequal to any DL signature symbol,*[4]
6. *I and O do not depend on concrete signature symbols, i.e. for a renaming ρ of signature symbols that maps individual/concept/role names to individual/concept/role names, we find $I(\rho(\alpha)) = \rho(I(\alpha))$ and $O(\rho(\alpha)) = \rho(O(\alpha))$ if $\rho(aux_i^\alpha) = aux_i^{\rho(\alpha)}$.*

We extend I to knowledge bases KB *by setting $I(\text{KB}) := \bigcup_{\beta\in\text{KB}} I(\beta)$ if $I(\beta)$ is defined for all $\beta \in$ KB and undefined otherwise. We extend I to sets of signature symbols S by setting $I(S) := \bigcup_{s\in S, I(s) \text{ defined}} I(s)$.*

K induces an entailment relation *$\vdash_K$ between knowledge bases* KB *and axioms α over a signature $\langle \mathbf{N_I}, \mathbf{N_C}, \mathbf{N_R}\rangle$, defined by setting* KB *$\vdash_K \alpha$ whenever I(KB) and $O(\alpha)$ are defined and $I(\text{KB}) \cup I(\mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R}) \cup P \models O(\alpha)$.*

We say that K is sound (complete) *if* KB *$\vdash_K \alpha$ implies (is implied by)* KB *$\models \alpha$ for all knowledge bases* KB *and axioms α for which I(KB) and $O(\alpha)$ are defined.*

[3] For instance, the calculus in [2] uses auxiliary statements $A \rightsquigarrow_R B$ for class names A and B.

[4] When clear from the context, we will generally omit α and simply write aux_i.

$C(a) \mapsto \{\texttt{subClass}(a, C)\}$	$R(a,b) \mapsto \{\texttt{subEx}(a, R, b, b)\}$	$a \in \mathbf{N_I} \mapsto \{\texttt{nom}(a)\}$
$\top \sqsubseteq C \mapsto \{\texttt{top}(C)\}$	$A \sqsubseteq \bot \mapsto \{\texttt{bot}(A)\}$	$A \in \mathbf{N_C} \mapsto \{\texttt{cls}(A)\}$
$\{a\} \sqsubseteq C \mapsto \{\texttt{subClass}(a, C)\}$	$A \sqsubseteq \{c\} \mapsto \{\texttt{subClass}(A, c)\}$	$R \in \mathbf{N_R} \mapsto \{\texttt{rol}(R)\}$
$A \sqsubseteq C \mapsto \{\texttt{subClass}(A, C)\}$	$A \sqcap B \sqsubseteq C \mapsto \{\texttt{subConj}(A, B, C)\}$	
$\exists R.\mathsf{Self} \sqsubseteq C \mapsto \{\texttt{subSelf}(R, C)\}$	$A \sqsubseteq \exists R.\mathsf{Self} \mapsto \{\texttt{supSelf}(A, R)\}$	
$\exists R.A \sqsubseteq C \mapsto \{\texttt{subEx}(R, A, C)\}$	$A \sqsubseteq \exists R.B \mapsto \{\texttt{supEx}(A, R, B, aux_1)\}$	
$R \sqsubseteq T \mapsto \{\texttt{subRole}(R, T)\}$	$R \circ S \sqsubseteq T \mapsto \{\texttt{subRChain}(R, S, T)\}$	
$R \sqsubseteq C \times D \mapsto \{\texttt{supProd}(R, C, D)\}$	$A \times B \sqsubseteq R \mapsto \{\texttt{subProd}(A, B, R)\}$	
$R \sqcap S \sqsubseteq T \mapsto \{\texttt{subRConj}(R, S, T)\}$		

Fig. 2. Input translation for K_{inst}

This definition further extends the above intuition of a materialisation calculus by explicitly introducing a datalog transformation I that is allowed to introduce arbitrarily many auxiliary constants aux_i^α. This can be utilised, for example, to perform a normalisation that introduces auxiliary concept names as part of the input translation. Yet, the input translation is limited in its expressivity, since it depends only on individual axioms and signature symbols. In particular, this precludes complex datalog translations as in [12,13]. Note that we do not make any assumptions on the computability or complexity of I and O, but both functions are typically very simple.

A noteworthy feature of materialisation calculi in the above sense is that they suggest materialisation-based (or consequence-driven) reasoning approaches: after translating a knowledge base to datalog facts, all consequences of these facts under the deduction rules can be computed in a bottom-up fashion, and (given that the function O is easy to compute, or even invertible) all supported entailments can be checked thereafter without further recursive computation. This contrasts with other reasoning principles such as the tableaux method where just a single entailment is checked in one run of the algorithm.

It is not hard to formulate the deduction algorithms presented for $\mathcal{EL}$-type logics in [2] and [5] as materialisation calculi in the sense of Definition 3. The calculus we present here, however, is derived from a datalog reduction that has originally been introduced in [10] for a rule language based on $\mathcal{EL}^{++}$. This approach can be modified to cover $\mathcal{SROEL}(\sqcap, \times)$ but does not directly yield a materialisation calculus in our sense since the set of datalog rules in [10] is not fixed but generated during the translation.

Theorem 1. *Consider the materialisation calculus $K_{inst} = \langle I_{inst}, P_{inst}, O_{inst}\rangle$ with I_{inst} defined as in Fig. 2, P_{inst} defined as in Fig. 3, and O_{inst} defined as $O_{inst}(C(a)) := \texttt{inst}(a, C)$ for $C \in \mathbf{N_C}$, $a \in \mathbf{N_I}$, and undefined otherwise. Then K_{inst} is sound and complete, i.e. it provides a materialisation calculus for instance checking for $\mathcal{SROEL}(\sqcap, \times)$ knowledge bases within which all axioms are normalised.*

A proof of this theorem will be presented below. It is not hard to obtain an intuition about the rules of P_{inst}. The IDB predicates `inst`, `triple`, and `self` correspond to ABox axioms for atomic concepts, roles, and concepts $\exists R.\mathsf{Self}$, respectively. Rule (1) serves as an initialisation rule that accounts for the first `inst` facts to be derived. Rule (2) specifies the (only) case where reflexive `triple` facts lead to `self` facts. The rules

(1)	$\texttt{nom}(x) \rightarrow \texttt{inst}(x,x)$
(2)	$\texttt{nom}(x) \wedge \texttt{triple}(x,v,x) \rightarrow \texttt{self}(x,v)$
(3)	$\texttt{top}(z) \wedge \texttt{inst}(x,z') \rightarrow \texttt{inst}(x,z)$
(4)	$\texttt{bot}(z) \wedge \texttt{inst}(u,z) \wedge \texttt{inst}(x,z') \wedge \texttt{cls}(y) \rightarrow \texttt{inst}(x,y)$
(5)	$\texttt{subClass}(y,z) \wedge \texttt{inst}(x,y) \rightarrow \texttt{inst}(x,z)$
(6)	$\texttt{subConj}(y_1,y_2,z) \wedge \texttt{inst}(x,y_1) \wedge \texttt{inst}(x,y_2) \rightarrow \texttt{inst}(x,z)$
(7)	$\texttt{subEx}(v,y,z) \wedge \texttt{triple}(x,v,x') \wedge \texttt{inst}(x',y) \rightarrow \texttt{inst}(x,z)$
(8)	$\texttt{subEx}(v,y,z) \wedge \texttt{self}(x,v) \wedge \texttt{inst}(x,y) \rightarrow \texttt{inst}(x,z)$
(9)	$\texttt{supEx}(y,v,z,x') \wedge \texttt{inst}(x,y) \rightarrow \texttt{triple}(x,v,x')$
(10)	$\texttt{supEx}(y,v,z,x') \wedge \texttt{inst}(x,y) \rightarrow \texttt{inst}(x',z)$
(11)	$\texttt{subSelf}(v,z) \wedge \texttt{self}(x,v) \rightarrow \texttt{inst}(x,z)$
(12)	$\texttt{supSelf}(y,v) \wedge \texttt{inst}(x,y) \rightarrow \texttt{self}(x,v)$
(13)	$\texttt{subRole}(v,w) \wedge \texttt{triple}(x,v,x') \rightarrow \texttt{triple}(x,w,x')$
(14)	$\texttt{subRole}(v,w) \wedge \texttt{self}(x,v) \rightarrow \texttt{self}(x,w)$
(15)	$\texttt{subRChain}(u,v,w) \wedge \texttt{triple}(x,u,x') \wedge \texttt{triple}(x',v,x'') \rightarrow \texttt{triple}(x,w,x'')$
(16)	$\texttt{subRChain}(u,v,w) \wedge \texttt{self}(x,u) \wedge \texttt{triple}(x,v,x') \rightarrow \texttt{triple}(x,w,x')$
(17)	$\texttt{subRChain}(u,v,w) \wedge \texttt{triple}(x,u,x') \wedge \texttt{self}(x',v) \rightarrow \texttt{triple}(x,w,x')$
(18)	$\texttt{subRChain}(u,v,w) \wedge \texttt{self}(x,u) \wedge \texttt{self}(x,v) \rightarrow \texttt{triple}(x,w,x)$
(19)	$\texttt{subRConj}(v_1,v_2,w) \wedge \texttt{triple}(x,v_1,x') \wedge \texttt{triple}(x,v_2,x') \rightarrow \texttt{triple}(x,w,x')$
(20)	$\texttt{subRConj}(v_1,v_2,w) \wedge \texttt{self}(x,v_1) \wedge \texttt{self}(x,v_2) \rightarrow \texttt{self}(x,w)$
(21)	$\texttt{subProd}(y_1,y_2,w) \wedge \texttt{inst}(x,y_1) \wedge \texttt{inst}(x',y_2) \rightarrow \texttt{triple}(x,w,x')$
(22)	$\texttt{subProd}(y_1,y_2,w) \wedge \texttt{inst}(x,y_1) \wedge \texttt{inst}(x,y_2) \rightarrow \texttt{self}(x,w)$
(23)	$\texttt{supProd}(v,z_1,z_2) \wedge \texttt{triple}(x,v,x') \rightarrow \texttt{inst}(x,z_1)$
(24)	$\texttt{supProd}(v,z_1,z_2) \wedge \texttt{self}(x,v) \rightarrow \texttt{inst}(x,z_1)$
(25)	$\texttt{supProd}(v,z_1,z_2) \wedge \texttt{triple}(x,v,x') \rightarrow \texttt{inst}(x',z_2)$
(26)	$\texttt{supProd}(v,z_1,z_2) \wedge \texttt{self}(x,v) \rightarrow \texttt{inst}(x,z_2)$
(27)	$\texttt{inst}(x,y) \wedge \texttt{nom}(y) \wedge \texttt{inst}(x,z) \rightarrow \texttt{inst}(y,z)$
(28)	$\texttt{inst}(x,y) \wedge \texttt{nom}(y) \wedge \texttt{inst}(y,z) \rightarrow \texttt{inst}(x,z)$
(29)	$\texttt{inst}(x,y) \wedge \texttt{nom}(y) \wedge \texttt{triple}(z,u,x) \rightarrow \texttt{triple}(z,u,y)$

Fig. 3. Deduction rules P_{inst}

(3) to (26) capture expected derivations for each of the axiom types as encoded by the EDB predicates. A special case is rule (4) which checks for global inconsistency. In implementations, such rules are typically not materialised since their effect can easily be taken into account when checking for entailments. Rules (9) and (10) make use of the auxiliary constants we use for handling existentials. Roughly speaking, each such constant represents the class of all role successors generated by the axiom from which it originates; Lemma 1 below formalises this intuition. The remaining rules (27) to (29) encode equality reasoning that is relevant in the presence of nominals. In particular, statements $\texttt{inst}(a,b)$ with a and b individuals encode equality of a and b. Most rules for axiomatising equality are not needed: the property that we require to hold is formalised in Lemma 2 below.

Axiom normalisation and the computations of I_{inst} and O_{inst} can be accomplished in linear time, and the time for reasoning in datalog is polynomial w.r.t. the size of the collection of ground facts. Together with the known P-hardness of $\mathcal{EL}^{++}$ [2], we obtain the following result, of which no formal proof seems to have been published so far:

Corollary 1. *Instance checking in $\mathcal{SROEL}(\sqcap, \times)$ and in OWL EL without datatype properties is* P *complete w.r.t. the size of the knowledge base.*

It is not hard to extend this result to OWL EL with datatype properties along the lines of datatype reasoning in $\mathcal{EL}^{++}$ [2], but this is not a direct consequence of the above theorem. The proof of Theorem 1 is established by Lemma 1 (soundness) and 3 (completeness) below. We start with the former since its proof provides some further intuition on the meaning of datalog atoms derived in the calculus. While soundness is easy to establish for most rules, the cases of rules (19) and (25) are slightly more intricate and make up most of the following proof.

Lemma 1. *For a $\mathcal{SROEL}(\sqcap, \times)$ knowledge base* KB *in normal form, a class name $D \in \mathbf{N_C}$, and an individual $a \in \mathbf{N_I}$, we find that* $\mathrm{KB} \vdash_{K_{inst}} D(a)$ *implies* $\mathrm{KB} \models D(a)$.

Proof. Let P be the datalog program $I_{\text{inst}}(\mathrm{KB}) \cup I_{\text{inst}}(\mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R}) \cup P_{\text{inst}}$. To interpret the IDB atoms that are derived by K_{inst}, we assign a concept expression $\kappa(c)$ to each constant c of P as follows:

- if $c \in \mathbf{N_I}$ then $\kappa(c) \coloneqq \{c\}$,
- if $c = aux_i^{\alpha}$ for $\alpha = A \sqsubseteq \exists R.B$, then $\kappa(c) \coloneqq B \sqcap \exists R^-.A$.

The concept used in the second case includes an inverse role, the semantics of which is defined by $(\exists R^-.A)^{\mathcal{I}} \coloneqq \{e \in \Delta^{\mathcal{I}} \mid \langle d, e\rangle \in R^{\mathcal{I}}, d \in A^{\mathcal{I}}\}$. Inverse roles are not supported by $\mathcal{SROEL}(\sqcap, \times)$ but are convenient for formulating this proof.

Now we can assign meaning to ground IDB atoms of P as follows:

- $\mathtt{inst}(c, A)$ with $A \in \mathbf{N_C}$: $\mathrm{KB} \models \kappa(c) \sqsubseteq A$,
- $\mathtt{inst}(c, d)$ with $d \in \mathbf{N_I}$: $\mathrm{KB} \models \kappa(c) \sqsubseteq \{d\}$,
- $\mathtt{triple}(c, R, d)$: $\mathrm{KB} \models \kappa(c) \sqsubseteq \exists R.\kappa(d)$,
- $\mathtt{self}(c, R)$: $\mathrm{KB} \models \kappa(c) \sqsubseteq \exists R.\mathsf{Self}$,

and in each case KB implies that $\kappa(c)$ is necessarily non-empty. Note that the second constant in any derived `inst` predicate must be in $\mathbf{N_C} \cup \mathbf{N_I}$, so the above definition covers all cases. We claim that an IDB atom is entailed by P only if the corresponding semantic conditions are satisfied by KB. In particular, this proves the overall claim.

We establish this claim by induction over the proof (tree) of an IDB atom. The claim clearly holds for the base case of rule (1) since nominals are necessarily non-empty. For almost all other rules, it is easy to apply the induction hypothesis immediately to the body atoms to obtain the desired conclusion in combination with the axioms of KB that the involved EDB atoms encode. This covers all rules but rule (19) and (25). The required non-emptiness of $\kappa(c)$ is easy to derive, and it is explicitly needed only for the conclusion in rules (21) and (22). Note how the precondition $\mathtt{inst}(x, z')$ serves to ensure non-emptiness of $\kappa(x)$ for rules (3) and (4). It thus remains to show the claim for the rules (19) and (25).

First consider the situation for rule (25). To establish the claim, we show that $P \models \mathtt{supProd}(R, C, D) \wedge \mathtt{triple}(c, R, d)$ implies $\mathrm{KB} \models \kappa(d) \sqsubseteq D$ (note that B is not a nominal based on our normal form). Non-emptiness of $\kappa(d)$ follows from the induction hypothesis on $P \models \mathtt{triple}(c, R, d)$. The assumptions imply $R \sqsubseteq C \times D \in \mathrm{KB}$. Yet, the

claim is obvious only if $d \in \mathbf{N_I}$. For the case $d \notin \mathbf{N_I}$, the statement cannot be concluded from the meanings provided for `triple` above.

Thus we assume that $d = aux_i^\alpha$ with $\alpha = A \sqsubseteq \exists V.B$ and we prove the following claim: if KB $\models R \sqsubseteq C' \times D$ and $P \models \texttt{triple}(c, R, d)$ then KB $\models \kappa(d) \sqsubseteq D$. We proceed by induction on the proof tree of $P \models \texttt{triple}(c, R, d)$:

- Rule (9). Then $R = V$ and the claim is obvious since $\kappa(d) = B \sqcap \exists R^-.A$ is a subclass of $\exists R^-.\top$.
- Rule (13). Then there is an axiom $R' \sqsubseteq R \in$ KB. Thus we obtain KB $\models R' \sqsubseteq C \times D$, to which we can apply the induction hypothesis to obtain KB $\models \kappa(d) \sqsubseteq D$.
- Rule (15). Then there is an axiom $R_1 \circ R_2 \sqsubseteq R \in$ KB. By the definition of $\mathcal{SROEL}(\sqcap, \times)$ knowledge bases, we find $\mathsf{ran}(R) \subseteq \mathsf{ran}(R_2)$, so KB contains axioms $R_2 \sqsubseteq S_1$, $S_1 \sqsubseteq S_2$, …, $S_{n-1} \sqsubseteq S_n$, and $S_n \sqsubseteq C' \times D \in$ KB. We conclude that KB $\models R_2 \sqsubseteq C' \times D$. Since the rule application requires $P \models \texttt{triple}(c', R_2, d)$, we obtain KB $\models \kappa(d) \sqsubseteq D$ by the induction hypothesis.
- Rule (16). This case is analogous to the case of rule (15).
- Rule (17). We obtain KB $\models R_2 \sqsubseteq C' \times D$ as in the case of rule (15). Since $P \models \texttt{self}(d, R_2)$, we obtain KB $\models \kappa(d) \sqsubseteq \exists R_2.\mathsf{Self}$ from the global induction hypothesis. Overall, we thus conclude KB $\models \kappa(d) \sqsubseteq D$.
- Rule (18). This case is analogous to the case of rule (17).
- Rule (19). Then there is an axiom $R_1 \sqcap R_2 \sqsubseteq R \in$ KB. By the definition of $\mathcal{SROEL}(\sqcap, \times)$ knowledge bases, we find $\mathsf{ran}(R) \subseteq \mathsf{ran}(R_1) \cup \mathsf{ran}(R_2)$, so there is $i \in \{1, 2\}$ such that KB contains axioms $R_i \sqsubseteq S_1$, $S_1 \sqsubseteq S_2$, …, $S_{n-1} \sqsubseteq S_n$, and $S_n \sqsubseteq C' \times D \in$ KB. We conclude that KB $\models R_i \sqsubseteq C' \times D$. Since the rule application requires $P \models \texttt{triple}(c, R_i, d)$, we obtain KB $\models \kappa(d) \sqsubseteq D$ by the induction hypothesis.
- Rule (21). Then there is an axiom $E \times F \sqsubseteq R \in$ KB. Since $P \models \texttt{inst}(c, E)$ and $P \models \texttt{inst}(d, F)$, we find that $\kappa(c)$ is non-empty and KB $\models \kappa(d) \sqsubseteq F$ by the global induction hypothesis. This shows that KB $\models \kappa(d) \sqsubseteq \exists R^-.\top$ so we conclude KB $\models \kappa(d) \sqsubseteq D$.

This concludes the inductive argument, since rule (29) is not relevant here since $d \notin \mathbf{N_I}$.

It remains to show that the claim of the main induction holds for applications of rule (19). To establish the claim, we show that $P \models \texttt{subRConj}(R, S, T) \wedge \texttt{triple}(c, R, d) \wedge \texttt{triple}(c, S, d)$ implies KB $\models \kappa(c) \sqsubseteq \exists T.\kappa(d)$. This is easy to see only for the case that $d \in \mathbf{N_I}$. For the case $d \notin \mathbf{N_I}$, the statement cannot be concluded from the meanings provided for `triple` above. To show that the claim holds in this case, assume $d = aux_i^\alpha$ with $\alpha = A \sqsubseteq \exists V.B$.

With these assumption, we consider the proof tree by which $P \models \texttt{triple}(c, T, d)$ is derived. We only require the "upper part" T_u of this proof tree that is inductively characterised as follows: T_u contains the root node (labelled with $\texttt{triple}(c, T, d)$); and if T_u contains a node labelled with an atom $\texttt{triple}(c, W, d)$ ($W \in \mathbf{N_R}$) then T_u also contains all of its child nodes. So the partial proof tree T_u may have IDB atoms as its leafs but has no leaf atoms of the form $\texttt{triple}(c, W, d)$. The root of T_u is derived with (a ground instance of) rule (19). We find that the premises $\texttt{triple}(c, R, d)$ and

$\texttt{triple}(c, S, d)$ of this rule can only be derived by rules (9), (13), (19), and (21). The rules (15) to (18) cannot occur since R and S are simple, and the rule (29) cannot occur since $d \notin \mathbf{N_I}$. Simplicity is propagated to roles in premises of rule (13) and (19), so we find that all rules applied throughout T_u are (9), (13), (19), or (21).

T_u traces back the derivation of atoms $\texttt{triple}(c, W, d)$. The base cases from which such atoms can be derived are only the rules (9) and (21). As a first case, consider the sub-proof tree of $\texttt{triple}(c, R, d)$ that T_u contains, and assume that this tree does not use rule (9). An easy induction shows that this implies $\text{KB} \models \kappa(c) \times \kappa(d) \sqsubseteq R$. Indeed, the remaining base case (21) is obvious, and the induction steps for rules (13) and (19) are easy based on the induction hypothesis. Now from the main induction hypothesis of the lemme, we obtain $\kappa(c) \sqsubseteq \exists S.\kappa(d)$ and $\kappa(c)$ non-empty. Together with $\kappa(c) \times \kappa(d) \sqsubseteq R$ and $R \sqcap S \sqsubseteq T \in \text{KB}$ this shows the claim $\text{KB} \models \kappa(c) \sqsubseteq \exists T.\kappa(d)$ and $\kappa(c)$ non-empty. An analogous argument is obtained if the proof for $\texttt{triple}(c, S, d)$ does not use rule (9).

It remains to establish the claim for the case that the proofs of both $\texttt{triple}(c, R, d)$ and $\texttt{triple}(c, S, d)$ involve some application of rule (9). For this case, we establish the following auxiliary claim (‡): if the subtree of T_u for deriving $\texttt{triple}(c, R, d)$ uses rule (9), and if $\text{KB} \models \kappa(c) \sqsubseteq \exists(V \sqcap W).\kappa(d)$ for some $W \in \mathbf{N_R}$, then $\text{KB} \models \kappa(c) \sqsubseteq \exists(V \sqcap W \sqcap R).\kappa(d)$ (where nested role conjunctions in concept expressions are allowed with the obvious semantics). Obviously, this statement also subsumes a similar claim using S instead of R. Intuitively speaking, (‡) states that role relations can be added conjunctively to the basic relation V along the proof of $\texttt{triple}(c, R, d)$. We show this claim by induction over the relevant rules occurring in the considered subtree of T_u:

- Rule (9). Then $R = V$ and the claim is immediate.
- Rule (13). Then there is an axiom $R' \sqsubseteq R \in \text{KB}$. By the induction hypothesis for (‡), we find that $\text{KB} \models \kappa(c) \sqsubseteq \exists(V \sqcap W \sqcap R').\kappa(d)$, and these statements clearly imply $\text{KB} \models \kappa(c) \sqsubseteq \exists(V \sqcap W \sqcap R' \sqcap R).\kappa(d)$ and thus (‡).
- Rule (19). Then there is an axiom $R_1 \sqcap R_2 \sqsubseteq R \in \text{KB}$. By the assumption, at least one of the proofs of the atoms $\texttt{triple}(c, R_i, d)$ in T_u uses rule (9). Assume without loss of generality that this is the case for R_1. By the hypothesis for (‡) we find $\text{KB} \models \kappa(c) \sqsubseteq \exists(V \sqcap W \sqcap R_1).\kappa(d)$.
  If the proof of $\texttt{triple}(c, R_2, d)$ does not use rule (9), we can apply our above reasoning for this case to obtain $\text{KB} \models \kappa(c) \times \kappa(d) \sqsubseteq R_2$, and we conclude $\text{KB} \models \kappa(c) \sqsubseteq \exists(V \sqcap W \sqcap R_1 \sqcap R_2).\kappa(d)$. Together with $R_1 \sqcap R_2 \sqsubseteq R \in \text{KB}$ we obtain $\text{KB} \models \kappa(c) \sqsubseteq \exists(V \sqcap W \sqcap R).\kappa(d)$ as required.
  If the proof of $\texttt{triple}(c, R_2, d)$ uses rule (9), then we can apply the induction hypothesis of (‡) to R_2 with the premise $\text{KB} \models \kappa(c) \sqsubseteq \exists(V \sqcap W \sqcap R_1).\kappa(d)$ to find $\text{KB} \models \kappa(c) \sqsubseteq \exists(V \sqcap W \sqcap R_1 \sqcap R_2).\kappa(d)$. Again, we may thus conclude $\text{KB} \models \kappa(c) \sqsubseteq \exists(V \sqcap W \sqcap R).\kappa(d)$ as required.
- Rule (21). This case is not relevant in the induction: no proof tree that satisfies the assumptions for (‡) uses this rule at its root (since it must use rule (9) somewhere), so it is never encountered in the inductive argument.

Now the overall claim is easily established. Since the derivation of $\texttt{triple}(c, R, d)$ uses rule (9) which can only entail triples for V (since this is the role occurring in d), we find $P \models \texttt{triple}(c, V, d)$. This entails $\text{KB} \models \kappa(c) \sqsubseteq \exists V.\kappa(d)$ and $\kappa(c)$ non-empty by the global induction hypothesis. Thus we can apply (‡) with $\text{KB} \models \kappa(c) \sqsubseteq \exists(V \sqcap V).\kappa(d)$ to

obtain KB $\models \kappa(c) \sqsubseteq \exists(V \sqcap R).\kappa(d)$. Applying (‡) a second time, we find KB $\models \kappa(c) \sqsubseteq \exists(V \sqcap R \sqcap S).\kappa(d)$. Together with $R \sqcap S \sqsubseteq T \in$ KB, this shows KB $\models \kappa(c) \sqsubseteq \exists T.\kappa(d)$ which establishes the claim. □

It remains to show the completeness of the calculus. Due to the presence of nominals, different constant symbols in the datalog program used by K_{inst} may represent the same description logic individuals. To take this into account, we define an equivalence relation on the Herbrand universe of such programs. To use this equivalence for identifying elements in a model, we must ensure that the logical properties of equivalent elements are the same, i.e. that the equivalence relation is a congruence in a certain sense. The following, slightly weaker property suffices in our case:

Lemma 2. *For a $\mathcal{SROEL}(\sqcap,\times)$ knowledge base* KB *in normal form, let P denote the datalog program $I_{inst}(\text{KB}) \cup I_{inst}(\mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R}) \cup P_{inst}$, and define an equivalence relation $\approx$ on the Herbrand universe of P to be the reflexive, symmetric, transitive closure of the relation $\{\langle c,d\rangle \mid P \models \texttt{inst}(c,d), d \in \mathbf{N_I}\}$.*

Given a constant c such that $c \approx a$ for some $a \in \mathbf{N_I}$, we find that $P \models \texttt{inst}(c,A)$ ($P \models \texttt{triple}(c,R,d)$, $P \models \texttt{triple}(d,R,c)$, $P \models \texttt{self}(c,R)$) implies $P \models \texttt{inst}(a,A)$ ($P \models \texttt{triple}(a,R,d)$, $P \models \texttt{triple}(d,R,a)$, $P \models \texttt{self}(a,R)$).

Proof. First note that, for every constant d with $P \models \texttt{inst}(d,b)$ and $b \in \mathbf{N_I}$, we find that $P \models \texttt{inst}(d,A)$ iff $P \models \texttt{inst}(b,A)$. This directly follows from the fact that $\mathcal{J}$ satisfies the rules (27) and (28). Given the preconditions of the claim, this already allows us to conclude that $P \models \texttt{inst}(c,A)$ iff $P \models \texttt{inst}(a,A)$ (†). This statement (†) subsumes the first part of the claim. Moreover, $P \models \texttt{inst}(a,a)$ and (†) imply $P \models \texttt{inst}(c,a)$. We thus can conclude that $P \models \texttt{triple}(d,R,c)$ implies $P \models \texttt{triple}(d,R,a)$ using rule (29).

To show the remaining cases of the claim, consider a consequence $\texttt{triple}(c,R,d)$ or $\texttt{self}(c,R)$ of P. We show the claim by induction over the structure of the proof (tree) of this consequence. First consider the possibilities for deriving $\texttt{triple}(c,R,d)$. The cases of rules (9) and (21) follow from (†). The cases of rules (13), (15) to (19), and (29) follow from the induction hypothesis. Now consider the possibilities for deriving $\texttt{self}(c,R)$. If rule (2) was used on a premise $\texttt{nom}(c) \wedge \texttt{triple}(c,R,c)$, we can conclude $\texttt{triple}(a,R,c)$ from the induction hypothesis, and $\texttt{triple}(a,R,a)$ from (†). The claim then follows since $\texttt{nom}(a)$ holds due to $a \in \mathbf{N_I}$. The cases of rules (14) and (20) follows directly from the induction hypothesis. The cases of rules (12) and (22) follow again from (†). □

We can now show the completeness of K_{inst}:

Lemma 3. *For a $\mathcal{SROEL}(\sqcap,\times)$ knowledge base* KB *in normal form, a class name $D \in \mathbf{N_C}$, and an individual $a \in \mathbf{N_I}$, we find that* KB $\models D(a)$ *implies* KB $\vdash_{K_{inst}} D(a)$.

Proof. Let P be the datalog program $I_{\text{inst}}(\text{KB}) \cup I_{\text{inst}}(\mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R}) \cup P_{\text{inst}}$. We show the contrapositive of the claimed implication. If KB $\nvdash_{K_{\text{inst}}} D(a)$, then $P \not\models O_{\text{inst}}(D(a))$. Then there is an Herbrand model $\mathcal{J}$ of P such that $\mathcal{J} \not\models O_{\text{inst}}(D(a))$. We provide a construction for a model $\mathcal{I}$ of KB such that $\mathcal{I} \not\models D(a)$, which shows that KB $\not\models D(a)$ as required.

Consider the equivalence relation $\approx$ as defined in Lemma 2, and let $[c] := \{d \mid d \approx c\}$ denote the $\approx$ equivalence class of c. Let Aux be the set of auxiliary constants of the form aux_i^α that occur in P. The domain of $\mathcal{I}$ is defined as

$$\Delta^{\mathcal{I}} := \{d_1, d_2 \mid d \in Aux, \mathcal{J} \models \mathtt{inst}(d, e) \text{ for some } e, d \not\approx a \text{ for all } a \in \mathbf{N_I}\} \cup \{[c] \mid c \in \mathbf{N_I}\}.$$

The indices 1 and 2 introduce two copies of each auxiliary constant $d \in Aux$; this is important to handle Self statements properly. For each element $e \in \Delta^{\mathcal{I}}$, we define a projection $\iota(e)$ to $\Delta^{\mathcal{J}}$ as follows: if e is of the form d_n then $\iota(e) := d$; if e is of the form $[c]$ then $\iota(e) := b$ for an arbitrary fixed $b \in [c]$. We can now define the interpretation function for $\mathcal{I}$. For each $c \in \mathbf{N_I}$, set $c^{\mathcal{I}} := [c]$. For each $A \in \mathbf{N_C}$, set $A^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \mathcal{J} \models \mathtt{inst}(\iota(d), A)\}$. For each $R \in \mathbf{N_R}$, we inductively define $R^{\mathcal{I}}$ to be the smallest set that contains the pairs $\langle d, d' \rangle \in R^{\mathcal{I}}$ for which one of the following conditions holds:

- $\mathcal{J} \models \mathtt{triple}(\iota(d), R, \iota(d'))$ and $d \neq d'$, or
- $\mathcal{J} \models \mathtt{self}(\iota(d), R)$ and $d = d'$,
- there is an axiom $S \sqsubseteq R \in$ KB and $\langle d, d' \rangle \in S^{\mathcal{I}}$,
- there is an axiom $S \circ T \sqsubseteq R \in$ KB and $\langle d, e \rangle \in S^{\mathcal{I}}$, $\langle e, d' \rangle \in T^{\mathcal{I}}$ for some $e \in \Delta^{\mathcal{I}}$,
- there is an axiom $A \times B \sqsubseteq R \in$ KB and $d \in A^{\mathcal{I}}$, $d' \in B^{\mathcal{I}}$.

Observe that all pairs $\langle d, d' \rangle \in R^{\mathcal{I}}$ are such that $\mathcal{J} \models \mathtt{triple}(\iota(d), R, \iota(d'))$ or $\mathcal{J} \models \mathtt{self}(\iota(d), R)$. This can be shown inductively for the last three cases of the above definition, since $\mathcal{J}$ satisfies rules (13) to (18), and (21) and (22).

Lemma 2 shows that the definition of $\mathcal{I}$ does not depend on the choice of $\iota([c]) \in [c]$. Since we assumed that $\mathcal{J} \not\models \mathtt{inst}(a, D)$, we find that $\mathcal{I} \not\models D(a)$. It remains to show that $\mathcal{I}$ is a model of KB. We consider all axiom types that may occur in KB, where we follow the cases of Fig. 2.

- $\top \sqsubseteq C$. Then $\mathcal{J} \models \mathtt{top}(C)$. For all $d \in \Delta^{\mathcal{I}}$, we find $\mathcal{J} \models \mathtt{inst}(\iota(d), e)$ for some e: this is required if $\iota(d) \in Aux$, and it follows from rule (1) if $\iota(d) \in \mathbf{N_I}$. Thus $\mathcal{J} \models \mathtt{inst}(\iota(d), C)$ by rule (3), and hence $d \in C^{\mathcal{I}}$ as required.
- $A \sqsubseteq \bot$. Then $\mathcal{J} \models \mathtt{bot}(A)$. If $d \in A^{\mathcal{I}}$, then $\mathcal{J} \models \mathtt{inst}(\iota(d), A)$. For a and D as in the main claim, we also find $\mathcal{J} \models \mathtt{cls}(D)$, and $\mathcal{J} \models \mathtt{inst}(a, a)$ (by rule (1)). Thus $\mathcal{J} \models \mathtt{inst}(a, D)$ by rule (4), contradicting our assumptions on $\mathcal{J}$. Thus $d \in A^{\mathcal{I}}$ cannot be, and $A^{\mathcal{I}} = \emptyset$ as required.
- $\{b\} \sqsubseteq C$. Then $\mathcal{J} \models \mathtt{subClass}(b, C)$ and $\mathcal{J} \models \mathtt{nom}(b)$. Thus $\mathcal{J} \models \mathtt{inst}(b, b)$ (1), and $\mathcal{J} \models \mathtt{inst}(b, C)$ (5). Then $b^{\mathcal{I}} = [b] \in C^{\mathcal{I}}$ follows from Lemma 2.
- $A \sqsubseteq \{c\}$. Then $\mathcal{J} \models \mathtt{subClass}(A, c)$. If $d \in A^{\mathcal{I}}$, then $\mathcal{J} \models \mathtt{inst}(\iota(d), A)$. By rule (5), we thus find $\mathcal{J} \models \mathtt{inst}(\iota(d), c)$ and thus $\iota(d) \approx c$ and $d = [c] = c^{\mathcal{I}}$ as required.
- $A \sqsubseteq C$. Then $\mathcal{J} \models \mathtt{subClass}(A, C)$. If $d \in A^{\mathcal{I}}$, then $\mathcal{J} \models \mathtt{inst}(\iota(d), A)$. By rule (5), we thus find $\mathcal{J} \models \mathtt{inst}(\iota(d), C)$ and thus $d \in C^{\mathcal{I}}$ as required.
- $A \sqcap B \sqsubseteq C$. Then $\mathcal{J} \models \mathtt{subConj}(A, B, C)$. If $d \in A^{\mathcal{I}} \cap B^{\mathcal{I}}$, then $\mathcal{J} \models \mathtt{inst}(\iota(d), A)$ and $\mathcal{J} \models \mathtt{inst}(\iota(d), B)$. By rule (6), we thus find $\mathcal{J} \models \mathtt{inst}(\iota(d), C)$ and thus $d \in C^{\mathcal{I}}$ as required.
- $\exists R.A \sqsubseteq C$. Then $\mathcal{J} \models \mathtt{subEx}(R, A, C)$. If $d \in (\exists R.A)^{\mathcal{I}}$ then there is $d' \in A^{\mathcal{I}}$ with $\langle d, d' \rangle \in R^{\mathcal{I}}$, hence $\mathcal{J} \models \mathtt{inst}(\iota(d'), A)$. Further, either $\mathcal{J} \models \mathtt{triple}(\iota(d), R, \iota(d'))$, or $d = d'$ and $\mathcal{J} \models \mathtt{self}(\iota(d), R)$. Thus $\mathcal{J} \models \mathtt{inst}(\iota(d), C)$ by rule (7) or (8), and thus $d \in C^{\mathcal{I}}$ as required.

– $A \sqsubseteq \exists R.B$. Then $\mathcal{J} \models \texttt{supEx}(A, R, B, aux_1)$. If $d \in A^{\mathcal{I}}$, then $\mathcal{J} \models \texttt{inst}(\iota(d), A)$. Thus $\mathcal{J} \models \texttt{triple}(\iota(d), R, aux_1)$ (9) and $\mathcal{J} \models \texttt{inst}(aux_1, B)$ (10). We require an element $d' \in \Delta^{\mathcal{I}}$ with $\iota(d') \approx aux_1$ and such that $\langle d, d' \rangle \in R^{\mathcal{I}}$ (the requirement $d' \in B^{\mathcal{I}}$ then follows from $\iota(d') \approx aux_1$, $\mathcal{J} \models \texttt{inst}(aux_1, B)$, and Lemma 2). If $aux_1 \approx c$ for some $c \in \mathbf{N_I}$, we can use $d' = [c]$. Indeed, $\langle d, d' \rangle \in R^{\mathcal{I}}$ can be concluded in this case: if $d = d'$, then $\mathcal{J} \models \texttt{triple}(c, R, c)$ by Lemma 2, and thus $\mathcal{J} \models \texttt{self}(c, R)$ by rule (2). Second, consider the case that $aux_1 \not\approx c$ for all $c \in \mathbf{N_I}$. Then there is some $n \in \{1, 2\}$ such that $d \neq (aux_1)_n$, so $d' = (aux_1)_n$ clearly satisfies the claim, even if $\iota(d) = \iota(d')$ should occur.

– $\exists R.\mathsf{Self} \sqsubseteq C$. Then $\mathcal{J} \models \texttt{subSelf}(R, C)$. Assume that $\langle d, d \rangle \in R^{\mathcal{I}}$. We show $\mathcal{J} \models \texttt{self}(\iota(d), R)$ by induction over the definition of $R^{\mathcal{I}}$, where the case of role chains is not relevant since R must be simple. The first case of the definition is impossible since $d = d$, and the second case is trivial. If $S \sqsubseteq R \in \mathrm{KB}$ and $\langle d, d \rangle \in S^{\mathcal{I}}$, then we have $\mathcal{J} \models \texttt{self}(\iota(d), S)$ (induction hypothesis) and the claim follows by rule (14). For the last case, assume $A \times B \sqsubseteq R$ and $d \in A^{\mathcal{I}} \cap B^{\mathcal{I}}$. Then $\mathcal{J} \models \texttt{inst}(\iota(d), A)$ and $\mathcal{J} \models \texttt{inst}(\iota(d), B)$, and the claim follows from rule (22).

  Thus $\mathcal{J} \models \texttt{self}(\iota(d), R)$ and we can apply rule (11) to obtain $\mathcal{J} \models \texttt{inst}(\iota(d), C)$, which shows $d \in C^{\mathcal{I}}$ as required.

– $A \sqsubseteq \exists R.\mathsf{Self}$. Then $\mathcal{J} \models \texttt{supSelf}(A, R)$. If $d \in A^{\mathcal{I}}$, then $\mathcal{J} \models \texttt{inst}(\iota(d), A)$. Thus $\mathcal{J} \models \texttt{self}(\iota(d), R)$ by rule (12). This shows $\langle d, d \rangle \in R^{\mathcal{I}}$ as required.

– $R \sqsubseteq T$, $R \circ S \sqsubseteq T$, $A \times B \sqsubseteq R$. Immediate from the definition of $T^{\mathcal{I}}$.

– $R \sqcap S \sqsubseteq T$. Then $\mathcal{J} \models \texttt{subRConj}(R, S, T)$. Assume $\langle d, e \rangle \in R^{\mathcal{I}} \cap S^{\mathcal{I}}$. As a first case, if $d = e$ then we can show $\mathcal{J} \models \texttt{self}(\iota(d), R)$ and $\mathcal{J} \models \texttt{self}(\iota(d), S)$ using the same inductive argument as in the case "$\exists R.\mathsf{Self} \sqsubseteq C$" above since R and S are simple. Thus we find that $\mathcal{J} \models \texttt{self}(\iota(d), T)$ by rule (20), and we conclude $\langle d, d \rangle \in T^{\mathcal{I}}$ as required.

  As a second case, if $d \neq e$, then we can show that $\mathcal{J} \models \texttt{triple}(\iota(d), R, \iota(e))$ (and $\mathcal{J} \models \texttt{triple}(\iota(d), S, \iota(e))$) using another inductive argument on the definition of $R^{\mathcal{I}}$ ($S^{\mathcal{I}}$). The first case of the definition is trivial, the second can be excluded. The third case is obtained from rule (13) using the induction hypothesis. The fourth case can be excluded since R (S) is simple, and the last case follows from rule (21). This completes the induction, and we can apply rule (19) to conclude $\mathcal{J} \models \texttt{triple}(\iota(d), T, \iota(e))$ from which we derive the required $\langle d, e \rangle \in T^{\mathcal{I}}$.

– $R \sqsubseteq C \times E$. Then $\mathcal{J} \models \texttt{supProd}(R, C, E)$. If $\langle d, e \rangle \in R^{\mathcal{I}}$ then we find either $\mathcal{J} \models \texttt{triple}(\iota(d), R, \iota(e))$, or $d = e$ and $\mathcal{J} \models \texttt{self}(\iota(d), R)$. Thus $\mathcal{J} \models \texttt{inst}(\iota(d), E)$ (by rule (23) or (24)) and $\mathcal{J} \models \texttt{inst}(\iota(e), E)$ (by rule (25) or (26)), and thus $d \in C^{\mathcal{I}}$ and $e \in E^{\mathcal{I}}$ as required.

– $C(b)$. Then $\mathcal{J} \models \texttt{subClass}(b, C)$, and we can apply the same arguments as for the case "$\{b\} \sqsubseteq C$" above.

– $R(b, c)$. Then $\mathcal{J} \models \texttt{supEx}(R, b, c, c)$. By rule (1), $\mathcal{J} \models \texttt{inst}(b, b)$. Thus $\mathcal{J} \models \texttt{triple}(b, R, c)$ (9). If $b \neq c$, then we find $\langle [b], [c] \rangle \in R^{\mathcal{I}}$ (using Lemma 2 to show $\mathcal{J} \models \texttt{triple}(\iota([b]), R, \iota([c]))$). Otherwise, if $b = c$, then $\mathcal{J} \models \texttt{self}(b, R)$ by rule (2), and we can also conclude $\langle [b], [c] \rangle \in R^{\mathcal{I}}$. □

4 Classification of $\mathcal{SROEL}(\sqcap, \times)$ Knowledge Bases

The calculus K_{inst} of Theorem 1 directly solves the instance checking problem for $\mathcal{SROEL}(\sqcap, \times)$. A materialisation calculus for checking satisfiability can easily be obtained by observing that an $\mathcal{SROEL}(\sqcap, \times)$ knowledge base is inconsistent if and only if K_{inst} infers a fact $\texttt{inst}(x, z)$ where $\texttt{bot}(z)$ holds. Another inference task is the computation of class subsumptions which is called *classification* when done for all atomic classes. Classification is the most important task in various typical applications of OWL EL, e.g. for the well-known ontology SNOMED CT that does not contain any individual names. In this section, we therefore study materialisation calculi for solving this inference problem for $\mathcal{SROEL}(\sqcap, \times)$.

As a standard inference problem of DL, class subsumption can be reduced to instance retrieval: to check a subsumption $A \sqsubseteq B$, one introduces a new individual name c and adds an assertion $A(c)$ to the knowledge base; then the subsumption holds if the modified knowledge base entails $B(c)$. Yet, the calculus K_{inst} cannot directly be adapted for subsumption checking. The reason is that the reduction to instance retrieval requires the knowledge base to be modified, leading to new entailments, possibly even to global inconsistency. Hence it is not feasible to introduce test individuals c for all (atomic) classes at load time so as to materialise all class subsumptions in parallel. Rather, one would have to use the calculus K_{inst} multiple times for computing subsumptions, where one run is needed for each subclass A to compute all entailments of the form $A \sqsubseteq B$. SNOMED CT, for example, has around 300,000 classes, and the same number of computation runs would be required to naively compute the subsumption hierarchy in this case.

Although this approach may seem to be rather inefficient, it allows us to obtain a sound and complete materialisation calculus for class subsumption in $\mathcal{SROEL}(\sqcap, \times)$, based on which algorithmic optimisations can then be developed for improving efficiency. To obtain this calculus, we simply "internalise" the various runs of K_{inst} based on assumptions of the form $A(c)$. To this end, we extend all IDB predicates with one additional parameter for storing a class name, with the underlying intuition that an according IDB fact of K_{inst} is entailed under the assumption that this class is non-empty. Note that the name of the constant c in the test assertion $A(c)$ is immaterial, so we will generally re-use the datalog constant A as the test instance of a class A.

Theorem 2. *Consider the materialisation calculus $K_{sc} = \langle I_{sc}, P_{sc}, O_{sc} \rangle$ with I_{sc} defined like I_{inst} in Fig. 2, and O_{sc} defined as $O_{sc}(A \sqsubseteq B) := \texttt{inst\_sc}(A, B, A)$ for $A, B \in \mathbf{N_C}$, and undefined otherwise. The program P_{sc} consists of the following rules:*

- *for each rule $r \in P_{inst}$ (Fig. 3), a rule r' obtained from r by adding a new body atom $\texttt{cls}(q)$, and replacing each IDB atom $\texttt{inst}(x, y)$ ($\texttt{triple}(x, y, z)$, $\texttt{self}(x, y)$) by an atom $\texttt{inst\_sc}(x, y, q)$ ($\texttt{triple\_sc}(x, y, z, q)$, $\texttt{self\_sc}(x, y, q)$), where q is a variable not occurring in r,*
- *the additional rule $(*)$ given by $\texttt{cls}(q) \rightarrow \texttt{inst\_sc}(q, q, q)$.*

Then K_{sc} is sound and complete, i.e. it provides a materialisation calculus for subsumption checking for $\mathcal{SROEL}(\sqcap, \times)$ knowledge bases within which all axioms are in normal form.

Proof. Immediate from the above discussion: the rule $\texttt{cls}(q) \rightarrow \texttt{inst\_sc}(q, q, q)$ states that each class name q is assumed to contain an individual q where this assumption is called "q" (third parameter). All other rules apply to arbitrary class names in the last parameter of IDB predicates. So $\texttt{inst\_sc}(x, y, q)$ can be derived iff the calculus K_{inst} entails $\texttt{inst\_sc}(x, y)$ using the additional assertion $\texttt{inst}(q, q)$. Clearly, this yields the same results as the assertion $q(c)$ for a new individual c (the assertion $\texttt{nom}(c)$ is not relevant in the deduction if c is new). So the claim follows from Theorem 1. □

We already noted that this materialisation calculus is not particularly efficient since deductions that are globally true are inferred under each local assumption q independently. Besides the redundancy in computation, this also means that the number of globally derived facts can multiply by the number of class names in the signature – more than 300,000 in the case of SNOMED. Our formalisation of materialisation calculi provides a direct measure of this increase: the maximal arity of IDB predicates in K_{sc} is four while it had been only three in K_{inst}, leading to potentially higher space requirements for storing the materialised derivations, and to increased upper bounds for the steps needed to check for individual entailments. It should be kept in mind that implementations may achieve lower runtime bounds by using suitable optimisations; yet many standard implementation techniques for datalog, including semi-naive materialisation and SLD resolution, are sensitive to the number of parameters in derived predicates. The arity of IDB predicates thus is an important measure for the efficiency of a materialisation calculus, and we will thus denote this parameter as the *arity of a calculus* and speak of binary/ternary/n-ary materialisation calculi.

The search for a more efficient materialisation calculus thus can be given a precise meaning: develop a ternary or binary calculus that is sound and complete for $\mathcal{SROEL}(\sqcap, \times)$ classification. As we will show in Section 5, such a calculus cannot exist. To illustrate that this is not clear *a priori*, we conclude this section by showing that there are materialisation calculi of lower arities for classification in fragments of $\mathcal{SROEL}(\sqcap, \times)$.

A binary materialisation calculus for classification in $\mathcal{ELH}$ – the $\mathcal{SROEL}(\sqcap, \times)$ fragment that only allows $\sqcap$, $\exists$, role and concept subsumptions – has been presented in [5]. Here, we present an extension of this calculus that supports Self, role conjunction, and concept products on the right of property inclusion axioms. We begin, however, by developing a ternary materialisation calculus that also supports role chains. The input translation can remain as in Fig. 2 but without the cases for $\top$, $\bot$, nominal classes, and concept products on the left-hand side of RIAs. The EDB predicates `top`, `bot`, and `subProd` are no longer used.

A set of IDB rules can be developed by restricting the rules of K_{sc} of Theorem 2. We use the numbers as in Fig. 3 for referring to the rules obtained from K_{inst}. Rules (3), (4), (21), and (22) are no longer needed due to the restriction of EDB predicates. In the absence of nominal classes, we find that all derivations $\texttt{inst\_sc}(x, y, q)$ are such that y is a DL class name, or y is a DL individual name and $x = y$. This is not hard to verify inductively by considering each rule, and the symbols used in relevant EDB facts. This shows that rules (27), (28), and (29) are obsolete as well. The essential feature of the remaining rule set is that the additional parameter q that has been introduced for K_{sc} above is no longer required for obtaining a sound and complete materialisation calculus:

Theorem 3. *Consider the materialisation calculus $K_{scc} = \langle I_{scc}, P_{scc}, O_{scc}\rangle$ with I_{scc} defined like I_{inst} in Fig. 2 but undefined for all axioms that use nominal classes, $\top$, $\bot$, or concept products on the left-hand side, and O_{scc} defined as $O_{scc}(A \sqsubseteq B) \coloneqq \texttt{inst}(A, B)$ if $A, B \in \mathbf{N_C}$, and undefined otherwise. The program P_{scc} consists of the rules* (1), (2), (5)–(20), *and* (23)–(26) *of Fig. 3 together with a new rule $\texttt{cls}(z) \rightarrow \texttt{inst}(z, z)$ $(*)$.*

Then K_{scc} is sound and complete, i.e. it provides a ternary materialisation calculus for subsumption checking for normalised $\mathcal{SROEL}(\sqcap, \times)$ knowledge bases that contain only $\sqcap$ (for concepts and roles), $\exists$, Self, $\circ$, and concept products on the right-hand side.

Proof. The above discussion already justifies the restriction to the selected rule set. It remains to be shown that the auxiliary parameter q used in K_{sc} is not needed here. To this end, let $K \coloneqq \langle I_{\text{scc}}, P, O_{\text{sc}}\rangle$ denote the materialisation calculus using the input function of K_{scc}, the output function of K_{sc}, and the rules P obtained from K_{scc} by modifying/extending derivation rules as for K_{sc} in Theorem 2 (equivalently, by deleting the rules (3), (4), (21), (22), (27)–(29) from P_{sc}). Let $\vdash$ denote the entailment relation of K_{scc}. We need to show that KB $\vdash \alpha$ iff KB $\vdash_K \alpha$.

The "if" direction can immediately be obtained from an easy induction to show that $\texttt{inst\_sc}(x, y, q)$/$\texttt{triple\_sc}(x, y, z, q)$/$\texttt{self\_sc}(x, y, q)$ is entailed by K only if $\texttt{inst}(x, y)$/$\texttt{triple}(x, y, z)$/$\texttt{self}(x, y)$ is entailed by K_{scc}. We detail the "only if" direction which is slightly more complicated. Consider a knowledge base KB, and let P' be the datalog program $I_{\text{sc}}(\text{KB}) \cup I_{\text{sc}}(\mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R}) \cup P_{\text{sc}}$. We first define a family of set $(Q(c))$ for constants c occurring in P'. Intuitively, $Q(c)$ contains exactly those class names q that may occur in the last parameter in IDB atoms with c in the first parameter that are entailed under K. We define the sets $Q(c)$ to be the smallest collection of sets such that:

- If $c \in \mathbf{N_C}$, then $Q(c) = \{c\}$.
- If $c \in \mathbf{N_I}$, then $Q(c) = \mathbf{N_C}$.
- If $c \in \mathbf{N_R}$, then $Q(c) = \emptyset$.
- If c is an auxiliary constant introduced in an EDB fact $\texttt{supEx}(A, R, B, c)$, then $Q(c) = \bigcup_{x \in P(c)} Q(x)$ where $P(c) \coloneqq \{x \mid \texttt{triple}(x, y, c)$ is entailed by $K_{\text{scc}}\}$.

Note that the mutual interdependencies between the sets $Q(c)$ are monotone, so it is indeed possible to find an assignment where all sets are minimal. Formally, we claim that whenever $\texttt{inst}(x, y)$/$\texttt{triple}(x, y, z)$/$\texttt{self}(x, y)$ is entailed by K_{scc} then we find that $\texttt{inst\_sc}(x, y, q)$/$\texttt{triple\_sc}(x, y, z, q)$/$\texttt{self\_sc}(x, y, q)$ is entailed by K for every $q \in Q(x)$. This can be shown inductively by considering all derivation rules.

If rules (1) or $(*)$ were used for deriving the fact in the condition of the claim, the claim follows immediately since $x \in Q(x)$ in both cases by definition. In rules (2), (5), (6), (8), (9), (11)–(14), (16), (18)–(20), (23), (24), and (26), the claim follows directly from the induction hypothesis, since the same first variable occurs in all IDB atoms in the rule.

For the remaining rules, first note that we have $Q(c) \supseteq \bigcup_{x \in P(c)} Q(x)$ for all constants c (†). For auxiliary constants, this follows directly from the definition of Q. For $c \in \mathbf{N_I}$ it is trivial. Moreover, it is easy to check that $P(c) = \emptyset$ if $c \in \mathbf{N_R}$ by noting that role names can occur only in derived facts of the form $\texttt{triple}(x, v, y)$ in the v position (as can be formalised by a simple induction). This establishes the claim for $c \in \mathbf{N_R}$. Similarly,

for $c \in \mathbf{N_C}$, we find $P(c) \subseteq \{c\}$. This is so, since K_{scc} entails facts $\texttt{triple}(x, v, c)$ with $c \in \mathbf{N_C}$ only if $c = x$, which can be readily seen by induction over the possible proof trees for facts using `triple`. Indeed, the only possible base cases for this induction are rules (9) and (18), where only the later can lead entail $\texttt{triple}(x, v, c)$ with $c \in \mathbf{N_C}$.

Returning to the main induction, we find that (†) implies $Q(x) \subseteq Q(x')$ for the rules (7), (15), and (17), so the claim for these cases follows from the induction hypothesis.

Now let $\texttt{inst}(x', z)$ be the fact derived by rule (10), and let $q \in Q(x')$. From the inductive definition of $Q(x')$, we conclude that there must be a chain of derived facts $\texttt{triple}(x_0, v_0, x_1), \ldots, \texttt{triple}(x_n, v_n, x')$ where $x_0 \in \mathbf{N_I} \cup \mathbf{N_C}$, all x_i for $i > 0$ are auxiliary individuals $x_i \notin \mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R}$, and $q \in Q(x_i)$ for all $i \geq 0$. Moreover, we can assume that all facts in this chain have been derived by rule (9). This follows since only rules (9) and (18) introduce new `triple` facts without requiring the prior existence of (chains of) such facts, and since facts derived by rule (18) can clearly be discarded from the chain. Thus, also the fact $\texttt{triple}(x_n, v_n, x')$ was derived by rule (9) and $q \in Q(x_n)$. Since the premises of rules (9) and (10) are the same, the derivation of $\texttt{triple}(x_n, v_n, x')$ can be changed into a derivation of $\texttt{inst}(x', z)$. The premise $\texttt{inst}(x_n, y')$ which is then used for applying rule (9) is such that $q \in Q(x_n)$. Applying our overall inductive argument to this premise shows that $\texttt{inst\_sc}(x_n, y', q)$ can be derived in K, so that also $\texttt{inst\_sc}(x', z, q)$ can indeed be derived.

Finally, consider rule (25) and some $q \in Q(x')$. If $x' \in \mathbf{N_I}$ then also $x \in \mathbf{N_I}$ since no other case is possible in the absence of nominals, so we find $q \in Q(x)$ as required for applying the induction hypothesis. For the case $x' \in \mathbf{N_C}$, we already argued in the proof of (†) above that $P(x') \subseteq \{x'\}$, so we find that $x = x'$ which shows the claim. Similarly, the case $x' \in \mathbf{N_R}$ can again be excluded.

As the last remaining case, assume that $x' \notin \mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R}$ is an auxiliary constant, let $\texttt{triple}(x, v, x')$ be the atom in the premise of the considered application of rule (25), and consider some $q \in Q(x')$. Using an inner induction on the derivation of $\texttt{triple}(x, v, x')$, we show the following claim: if $z_2 \in \mathsf{ran}(v)$ and K_{scc} derives $\texttt{triple}(x, v, x')$, then K derives $\texttt{inst\_sc}(x', z_2, q)$. Clearly, this subsumes the overall claim for rule (25) as a special case.

First assume that $\texttt{triple}(x, v, x')$ was derived using rule (9). Using an argument as for the case of rule (10) above, we find a chain $\texttt{triple}(x_0, v_0, x_1), \ldots, \texttt{triple}(x_n, v_n, x')$ of facts which are derived by rule (9), and where $q \in Q(x_i)$ for all $i \geq 0$. Since x' uniquely determines the role name that is used in rule (9), we find that $v = v_n$, so rule (25) is applicable to $\texttt{triple}(x_n, v, x')$ for which we already have shown $q \in Q(x_n)$. Since $z_2 \in Q(v)$ there are axioms $v \sqsubseteq v_1, \ldots, v_{n-1} \sqsubseteq v_n$ and $v_n \sqsubseteq C \times z_2$ in KB. Due to rule (13) K_{scc} derives $\texttt{triple}(x_n, v_n, x')$. By the global induction hypothesis, K derives $\texttt{triple}(x_n, v_n, x', q)$, so the claim follows by (25).

If $\texttt{triple}(x, v, x')$ was derived based on an axiom $v' \sqsubseteq v$ using rule (13), then we find $z_2 \in \mathsf{ran}(v')$, so the claim follows by the inner induction. Analogously, if rule (15) or (16) was used for an axiom $v_1 \circ v_2 \sqsubseteq v$, then $z_2 \in \mathsf{ran}(v_2)$ due to the structural restrictions on axioms of the form $v \sqsubseteq z_1 \times z_2$ in $\mathcal{SROEL}(\sqcap, \times)$ knowledge bases. The case for rule (19) is also analogous.

(1)	$\texttt{nom}(x) \rightarrow \texttt{inst}(x, x)$
(2)	$\texttt{cls}(y) \rightarrow \texttt{inst}(y, y)$
(3)	$\texttt{rol}(v) \rightarrow \texttt{srole}(v, v)$
(4)	$\texttt{supEx}(y, v, z, x') \rightarrow \texttt{inst}(x', z)$
(5)	$\texttt{nom}(x) \wedge \texttt{supEx}(y, v, z, x') \wedge \texttt{inst}(x, y) \wedge \texttt{inst}(x', x) \rightarrow \texttt{self}(x, v)$
(6)	$\texttt{subClass}(y, z) \wedge \texttt{inst}(x, y) \rightarrow \texttt{inst}(x, z)$
(7)	$\texttt{subConj}(y_1, y_2, z) \wedge \texttt{inst}(x, y_1) \wedge \texttt{inst}(x, y_2) \rightarrow \texttt{inst}(x, z)$
(8)	$\texttt{supEx}(y, v, z, x') \wedge \texttt{subEx}(v', y', z') \wedge \texttt{inst}(x, y) \wedge \texttt{srole}(v, v') \wedge \texttt{inst}(x', y') \rightarrow \texttt{inst}(x, z')$
(9)	$\texttt{subEx}(v, y, z) \wedge \texttt{self}(x, v) \wedge \texttt{inst}(x, y) \rightarrow \texttt{inst}(x, z)$
(10)	$\texttt{subSelf}(v, z) \wedge \texttt{self}(x, v) \rightarrow \texttt{inst}(x, z)$
(11)	$\texttt{supSelf}(y, v) \wedge \texttt{inst}(x, y) \rightarrow \texttt{self}(x, v)$
(12)	$\texttt{subRole}(v, w) \wedge \texttt{srole}(u, v) \rightarrow \texttt{srole}(u, w)$
(13)	$\texttt{subRole}(v, w) \wedge \texttt{self}(x, v) \rightarrow \texttt{self}(x, w)$
(14)	$\texttt{subRConj}(v_1, v_2, w) \wedge \texttt{srole}(u, v_1) \wedge \texttt{srole}(u, v_2) \rightarrow \texttt{srole}(u, w)$
(15)	$\texttt{subRConj}(u, v, w) \wedge \texttt{self}(x, u) \wedge \texttt{self}(x, v) \rightarrow \texttt{self}(x, w)$
(16)	$\texttt{supProd}(v, z_1, z_2) \wedge \texttt{supEx}(y, u, z, x') \wedge \texttt{inst}(x, y) \wedge \texttt{srole}(u, v) \rightarrow \texttt{inst}(x, z_1)$
(17)	$\texttt{supProd}(v, z_1, z_2) \wedge \texttt{self}(x, v) \rightarrow \texttt{inst}(x, z_1)$
(18)	$\texttt{supProd}(v, z_1, z_2) \wedge \texttt{supEx}(y, u, z, x') \wedge \texttt{srole}(u, v) \rightarrow \texttt{inst}(x', z_2)$
(19)	$\texttt{supProd}(v, z_1, z_2) \wedge \texttt{self}(x, v) \rightarrow \texttt{inst}(x, z_2)$

Fig. 4. Deduction rules $P_{\text{sc-}}$

If (17) or (18) was used with $v_1 \circ v_2 \sqsubseteq v$, then $x = x'$ and, as above, $z_2 \in \mathsf{ran}(v_2)$. So there are axioms $v_2 \sqsubseteq v_2^1, \ldots, v_2^{n-1} \sqsubseteq v_2^n$ and $v_2^n \sqsubseteq C \times z_2$ in the knowledge base. Using rules (14) and (26) suffices to establish the claim since $Q(x) = Q(x')$. □

In terms of OWL 2, the DL of the previous theorem covers all OWL EL ontologies without datatype properties and the constructs `owl:Thing`, `owl:topObjectProperty`, `owl:Nothing`, `owl:bottomObjectProperty`, `objectHasValue` and `objectOneOf`. Note how the reasoning used for obtaining this calculus exploits our datalog-based description of materialisation calculi, which allows us to conveniently argue about possible derivations.

It is not hard to further simplify K_{scc} for the case that no role chains occur in the knowledge base. The main observation here is that `triple` atoms then can only be inferred by existential quantifiers on the right-hand side, and by applying role hierarchies and role conjunctions. The effect of the latter two features can be integrated using a new predicate that encodes inferred subrole relations, using rules similar to the ones for subclass and class conjunction. With this change, a statement $\texttt{triple}(x, w, z')$ is equivalent to the conjunction $\texttt{subEx}(y, v, z, x') \wedge \texttt{inst}(x, y) \wedge \texttt{srole}(v, w) \wedge \texttt{inst}(x', z')$, and we can use this conjunction in all rules that include `triple` in their bodies. Some further simplifications apply if not all arguments of `triple` are relevant to some rule application. Note that rule (10) of Fig. 3 is still needed, but since we consider a case where all classes can be assumed to be non-empty (i.e. where $\texttt{inst}(y, y)$ holds for all classes y), we can simplify this rule as well. We thus obtain the following materialisation calculus that extends the binary classification calculus for $\mathcal{ELH}$ as presented in [5].

Theorem 4. *Consider the materialisation calculus $K_{\text{sc-}} = \langle I_{\text{sc-}}, P_{\text{sc-}}, O_{\text{sc-}} \rangle$ with $I_{\text{sc-}}$ defined like I_{inst} in Fig. 2 but undefined for all axioms that use nominal classes, $\top$, $\bot$, role*

chains $\circ$, or concept products on the left-hand side, $P_{sc\text{-}}$ defined as in Fig. 4, and $O_{sc\text{-}}$ defined as $O_{sc\text{-}}(A \sqsubseteq B) := \texttt{inst}(A, B)$ if $A, B \in \mathbf{N_C}$, and undefined otherwise.

Then $K_{sc\text{-}}$ is sound and complete, i.e. it provides a binary materialisation calculus for subsumption checking for normalised $\mathcal{SROEL}(\sqcap, \times)$ knowledge bases that contain only $\sqcap$ (for concepts and roles), $\exists$, Self*, and concept products on the right-hand side.*

Note that the DL of the previous theorem still is a significant extension of $\mathcal{ELH}$ since it allows role conjunction (and thus role disjointness), local reflexivity (Self), and EL-admissible range restrictions (expressed as concept products on the right).

5 Minimal Arities of Materialisation Calculi

The materialisation calculi we discussed for reasoning in $\mathcal{SROEL}(\sqcap, \times)$ above featured various arities: while some reasoning problems could be solved by binary and ternary calculi, the presented classification calculus for $\mathcal{SROEL}(\sqcap, \times)$ is 4-ary. Higher arities lead to increased upper bounds on the number of consequences that are inferred, and hence increase the potential space requirement and computation time for materialisation-based implementations. It is therefore desirable to develop materialisation calculi of minimal arity. In this section, we establish lower bounds on the arity of materialisation calculi for various reasoning problems.

Our general proof strategy is as follows. For a contradiction, we suppose that there is a materialisation calculus of lower arity that solves a given reasoning problem. We then consider a particular instance of that problem, given by a knowledge base KB from which a relevant consequence α must follow. Since the calculus is assumed to be complete, we obtain an according datalog derivation with a corresponding proof tree. This proof tree is then modified by renaming constants, leading to a variant of the proof tree that is still valid for the given materialisation calculus, but that is based on different (renamed) assumptions. The modified assumptions correspond to a modified knowledge base KB′, and by our construction we find that the materialisation calculus still computes the entailment of α on the input KB′. We then show that α is not entailed by KB′, so that the calculus is proven to be unsound. Since KB′ is based on the modified proof tree, some graph theoretic arguments are required to establish this last step.

A central notion of this proof strategy is the following modification of proof trees.

Definition 4. *Consider a materialisation calculus $K = \langle I, P, O\rangle$ and a knowledge base* KB *such that I*(KB) *is defined, and a proof tree $T = \langle N, E, \lambda\rangle$ for I*(KB) $\cup$ $I(\mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R}) \cup P$*. We say that a DL signature symbol σ* occurs *in a ground atom F if F contains σ as a constant, or if F contains some auxiliary constant aux_i^α such that σ occurs in α. The* interface *of a node $n \in N$ is the set of signature symbols that occur in $\lambda(n)$.*

The (labels of) T can be diversified *by the following recursive construction:*

- *replace all signature symbols s that do not occur in the interface of the root node by a fresh symbol s' that has not yet been used in T or in this construction,*
- *recursively diversify the subtrees below each of the direct child nodes of the root.*

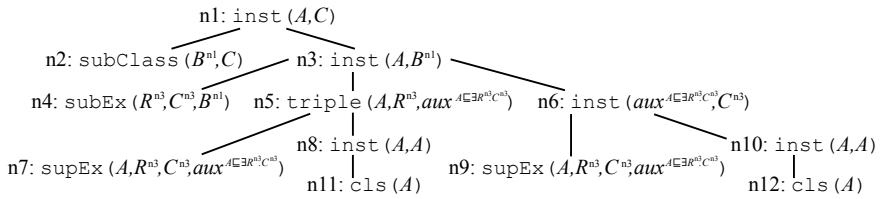

Fig. 5. Diversification of a K_{scc} proof for $\{A \sqsubseteq \exists R.C, \exists R.C \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$

We tacitly assume that the datalog signature contains all required new constant names. Note that the renaming may affect auxiliary constants by renaming symbols in the axioms that are part of their name. The diversification is thus obtained by replacing some signature symbols with fresh symbols. This replacement may not be uniform throughout the tree, and we use s^n to denote the symbol by which s is replaced in node n.

Intuitively speaking, the above renaming removes any re-use of constant names throughout the proof tree that is not strictly necessary for applying the rules of P. What is "strictly necessary" is captured by the *interface* of each node: constants that are not in the interface of a rule application can be renamed uniformly in all descendants of the current node without affecting the correctness of the proof tree. This creates a direct connection between the arity of a calculus and the amount of renaming that can be accomplished when diversifying a proof tree.

Figure 5 shows an example diversification based on the calculus K_{scc} of Theorem 3, where we use the notation from Definition 4 for denoting renamed symbols. Note how C is renamed to C^{n3} in some but not in all labels. Also note that no further renamings occur below the nodes $n5$ and $n6$ since all relevant symbols occur in their interface due to the auxiliary constant. As expected, the diversification is again a proof tree for a knowledge base that contains suitably renamed axioms:

The diversification corresponds to a proof tree for a knowledge base that contains renamed axioms:

Definition 5. *Consider a materialisation calculus K, knowledge base* KB*, and proof tree T as in Definition 4. Let λ' denote a diversified labelling for T.*

Let $m \in N$ be a leaf node with $\lambda(m) \in I(\alpha)$ for some $\alpha \in$ KB. By Definition 3, one can rename symbols in α to obtain an axiom α' such that $\lambda'(m) \in I(\alpha')$. Concretely, α' is obtained from α be replacing all symbols s in the interface of m by s^m, and by replacing all other symbols t by some fresh symbol t' not used anywhere yet. We select one such axiom α'_m for each such node m.

The diversification KB′ *of* KB *is the knowledge base* $\text{KB}' := \{\alpha'_n \mid n \in N, n \text{ a leaf}\}$. *The tree structure of T can be used to represent* KB′ *as a set of nested sets Γ_n for $n \in N$, recursively defined by setting $\Gamma_n := \{\alpha'_m \mid \langle n, m\rangle \in E, m \text{ a leaf}\} \cup \{\Gamma_m \mid \langle n, m\rangle \in E, m \text{ not a leaf}\}$. We say that an axiom or set is* below *a set Γ_n if it is either an element of Γ_n, or if it is (recursively) below some element of Γ_n.*

For Fig. 5, the diversified knowledge base is $\{A \sqsubseteq \exists R^{n3}.C^{n3}, \exists R^{n3}.C^{n3} \sqsubseteq B^{n1}, B^{n1} \sqsubseteq C\}$ and we have $\Gamma_{n1} = \{B^{n1} \sqsubseteq C, \{\exists R^{n3}.C^{n3} \sqsubseteq B^{n1}, \{A \sqsubseteq \exists R^{n3}.C^{n3}\}\}\}$. Since the underlying calculus is correct, the conclusion still follows from the diversified knowledge

base, and the diversified proof tree is still correct. Below we use diversification to construct proof trees with invalid conclusions for calculi with insufficient arities. But first let us note that diversification does indeed always lead to a new proof tree that can be constructed based on the diversified knowledge base.

Lemma 4. *Consider a materialisation calculus K, knowledge base* KB*, and proof tree T as in Definition 4. The diversification T' of T is a proof tree for $I(\mathrm{KB}') \cup I(\mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R}) \cup P$ where* KB′ *is the diversification of* KB*.*

Proof. It is easy to see that the conditions on proof trees are satisfied for all non-leaf nodes of T'. All leafs of T are labelled with EDB atoms of $I(\mathrm{KB}) \cup I(\mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R})$. Since I is assumed to be independent of concrete signature symbols (Definition 3), we find that for all EDB atoms $\alpha \in I(\mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R})$ there is a corresponding atom $\rho(\alpha) \in I(\mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R})$ for any renaming ρ, since all symbols used by renamings are part of the signature. Finally, if $m \in N$ is a leaf node with $\lambda(m) \in I(\alpha)$ for some $\alpha \in \mathrm{KB}$, then by Def. 5 there is $\alpha'_m \in \mathrm{KB}'$ such that $\lambda'(n) \in I(\alpha'_n)$. □

By the construction in Definition 5, if l is the maximal number of premises in rules of K, then each set Γ_n has at most l elements. Moreover, if $\Gamma_m \in \Gamma_n$, then the DL signature symbols that occur in axioms below Γ_m either belong to the interface of n, or occur only in axioms of KB′ that are below Γ_m. Since the interface includes all DL symbols that occur in the ground datalog atom that is derived at a certain node of the proof tree, the use of auxiliary constants can require the inclusion of *all* symbols of a given input axiom into the interface. Yet, the arity clearly limits the number of axioms for which this may be the case: for a calculus of arity a, the interface of any node can comprise no more than the set of DL symbols that occur in a axioms of the input knowledge base. These observations can also be interpreted graphically based on the *dependency graph* of KB′ which encodes the sharing of signature symbols in axioms:

Definition 6. *The* dependency graph *of a knowledge base* KB *is the graph that has the signature symbols in* KB *as its nodes, and, for each axiom of* KB *that has exactly n signature symbols, an n-ary hyperedge connecting these n symbols.*

The sets of axioms Γ_n can be viewed as subgraphs of a dependency graph, where the interface of the node n describes the nodes that this subgraph is allowed to share with the remaining graph. In our subsequent proofs, we will argue that the interface that a calculus of a certain arity allows a node to have is necessarily too small for establishing this connection between such a subgraph and the rest of the given dependency graph. For this argument to work, it is important that the subgraphs described by axioms below a set Γ_n can be assumed to be sufficiently distinct from the rest. For example, subgraphs Γ_n that contain only single axioms can always be connected if an auxiliary constant for this axiom occurs in $\lambda(n)$; conversely, the same applies to subgraphs that contain all but a single axiom. Since axioms in Γ_n play the role of leafs in the underlying proof tree, the next lemma helps us to find subgraphs Γ_n that have the right amount of axioms:

Lemma 5. *Consider a finite tree structure T with nodes N and edges E such that each node of T has at most k children. Let Γ_n denote the set of leafs of the subtree of T which has $n \in N$ as its root, and let l denote the total number of leafs in T. For any number*

$d > 1$ with $l > (k+1)(d-1)$, there must be a node $n \in N$ such that Γ_n contains at least d and at most $l - d$ elements.

Proof. Assume that there is a node $m \in N$ for which Γ_n contains at least $l - d + 1$ elements, but for which all children m' of m are such that $\Gamma_{n'} \leq d - 1$. Then Γ_n can contain at most $k(d-1)$ elements. Therefore, a node like m can only exist if $l - d + 1 \leq k(d-1)$, i.e. if $l \leq (k+1)(d-1)$. The precondition of the claim states that this is not the case. Hence every node n_0 of T with at least $l - d + 1$ leafs below it has a child node n_1 with $|\Gamma_{n_1}| > d - 1$. If $|\Gamma_{n_1}| > l - d + 1$, there is a child n_2 of n_1 with $|\Gamma_{n_2}| > d - 1$. Since T does not contain an infinite chain $n_0, n_1, n_2 m \ldots$, this construction ends with a node n as required in the claim. The overall claim follows since T contains a node with more than $l - d + 1$ leafs below it: its root. □

As a first application of this machinery, we show a result for DLs that are much weaker than $\mathcal{SROEL}(\sqcap, \times)$.

Theorem 5. *Let $\mathcal{L}$ be a DL with GCIs, existential quantification, and role chains. Every materialisation calculus that is sound and complete for classification or instance retrieval in $\mathcal{L}$ has arity three or more.*

Proof. We first consider the classification problem. For a contradiction, suppose that there is a binary materialisation calculus $K = \langle I, P, O \rangle$ such that I is defined on all axioms of $\mathcal{L}$, and O is defined for all axioms of the form $A \sqsubseteq B$. Let l be the maximal number of body atoms in any rule of P, and consider some $k > 2(l+1)$. A knowledge base KB is defined to contain the following axioms:

- $D_i \sqsubseteq \exists S_i.D_{i+1}$ for all $i \in \{0, \ldots, k\}$,
- $D_{k+1} \sqsubseteq \exists R_{k+1}.B$,
- $S_i \circ R_{i+1} \sqsubseteq R_i$ for all $i \in \{0, \ldots, k\}$,
- $\exists R_0.B \sqsubseteq B$.

Then KB entails $D_0 \sqsubseteq B$. Thus, by the assumption on K, there is a proof tree T for deriving $O(D_0 \sqsubseteq B)$ for the program $I(\mathrm{KB}) \cup I(\mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R}) \cup P$. Let $T' = \langle N, E, \lambda' \rangle$ be the diversified proof tree obtained from T by using renamed symbols s^n as in Definition 4, and let KB′ be the according diversified knowledge base.

We claim that KB′ has a model $\mathcal{I}$ that does not satisfy $D_0 \sqsubseteq B$. Define $\Delta^{\mathcal{I}} := \{D_i^n \mid 0 \leq i \leq k+1, n \in N\} \cup \{B^n \mid n \in N\}$. The interpretation for role and concept names is defined as follows, where $n, m \in N$ are nodes of T', and d_i, r_i, s_i, b denote arbitrary signature symbols where the names are chosen to hint at the actual symbols they may represent based on the given conditions:

- $(D_i^n)^{\mathcal{I}} := \{D_i^n\}$ for all $i = 0, \ldots, k+1$,
- $(S_i^n)^{\mathcal{I}} := \{\langle d_i, d_{i+1} \rangle \mid d_i \sqsubseteq \exists S_i^n.d_{i+1} \in \mathrm{KB}'\}$,
- $(R_{k+1}^n)^{\mathcal{I}} := \{\langle d_{k+1}, b \rangle \mid d_{k+1} \sqsubseteq \exists R_{k+1}^n.b \in \mathrm{KB}'\}$,
- for $i = k+1, \ldots, 1$ recursively define $(R_{i-1}^n)^{\mathcal{I}} := \{\langle d_{i-1}, b \rangle \mid s_{i-1} \circ r_i \sqsubseteq r_{i-1} \in \mathrm{KB}', \langle d_{i-1}, d_i \rangle \in s_{i-1}^{\mathcal{I}}, \langle d_i, b \rangle \in r_i^{\mathcal{I}}\}$,
- $(B^n)^{\mathcal{I}} := \{B^n\} \cup \{d \in \Delta^{\mathcal{I}} \mid \langle d, B^n \rangle \in r_0^{\mathcal{I}} \text{ and } \exists r_0.B^n \sqsubseteq B^n \in \mathrm{KB}'\}$.

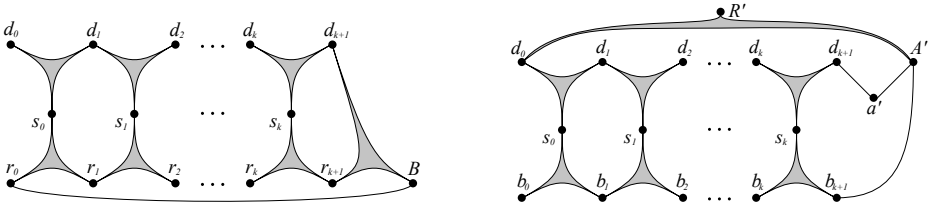

Fig. 6. Dependency graphs for the proofs of Theorem 5 (left) and 6 (right)

It is easy to see that $\mathcal{I}$ is a model of KB′. It remains to show that $D_0^{\mathcal{I}} \not\subseteq B^{\mathcal{I}}$. By the construction of $\mathcal{I}$, we see that $D_0 \in B^{\mathcal{I}}$ only if KB′ contains the following axioms:

- chains of axioms $d_0 \sqsubseteq s_0.d_1, \ldots, d_k \sqsubseteq s_k.d_{k+1}$ and $s_0 \circ r_1 \sqsubseteq r_0, \ldots, s_k \circ r_{k+1} \sqsubseteq r_k$ where we find $d_0 = D_0$, $d_i = D_i^o$ for some $o \in N$, $s_i = S_i^o$ for some $o \in N$, and $r_i = R_i^o$ for some $o \in N$,
- $d_{k+1} \sqsubseteq B$,
- $\exists r_0.B \sqsubseteq B$.

For a contradiction, suppose that KB′ contains a set KB″ of axioms of the above form. The dependency graph of KB″ is depicted in Fig. 6. Based on the construction of KB′, the axioms of KB″ must be distributed over sets $(\Gamma_o)_{o \in N}$ as in Definition 5. By Lemma 5 and our choice of k, there is a node $o \in N$ such that Γ_o has three axioms of the form $d_i \sqsubseteq \exists s_i.d_{i+1}$ below it, and such that three other axioms of this form are not below it (to obtain this from Lemma 5, we simply consider the subtree of T' that is induced by removing all leafs that do not correspond to axioms of this form).

Now Γ_o induces a subgraph of the dependency graph of KB″. Since K has arity 2, the maximal amount of symbols in the interface of o is the set of symbols used in two axioms of KB″ (the interface may also contain less symbols, e.g. two individual symbols if no auxiliary constants are used, but this case is subsumed). It is not hard to see that the subgraph induced by Γ_o must share more symbols with the rest of the dependency graph. First note that the interface of o must contain at least one symbol d_i with $i < k + 1$ since some axiom involving such a symbol does not occur below Γ_o.

Moreover, there are three axioms of the form $d_i \sqsubseteq \exists s_i.d_{i+1}$ below Γ_o, but not all of the involved three symbols s_i can be in the interface of o (since no two of them belong to the same axiom of KB″). Thus, an axiom of the form $s_i \circ r_{i+1} \sqsubseteq r_i$ must be below Γ_o. But the dependency graph of KB′ contains a cycle through the nodes $r_0, r_1, \ldots, r_{k+1}, B, r_0$. The interface of Γ_o can contain two nodes from this cycle only if they belong to the same axiom in KB″ (since the interface already contains a node d_i which certainly is not in any axiom together with some node in the cycle). Thus at most one axiom from this chain could not be below Γ_o, while all other axioms must be below Γ_o. But by construction three axioms of the form $d_i \sqsubseteq \exists s_i.d_{i+1}$ are not below Γ_o, so the involved symbols s_i must belong to the interface of o whenever the according axiom $s_i \circ r_{i+1} \sqsubseteq r_i$ is below Γ_o. Hence this is the case for at least two symbols s_i which do not occur together in an axiom, and which do not occur in an axiom that involves the symbol d_i that we reasoned to be part of the interface as well. Thus, o must have symbols from three distinct axioms in its interface, which cannot be.

Summing up, Γ_o cannot exist, and thus KB$''$ cannot be contained in KB$'$. So $\mathcal{I}$ does not satisfy $D_0 \sqsubseteq B$, and thus the latter is not a consequence of KB$'$. Since T' is a proof tree for $I(\text{KB}') \cup I(\mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R}) \cup P$ by Lemma 4, K derives $D_0 \sqsubseteq B$. So K cannot be sound, contradicting our assumption of its existence.

The result for instance retrieval is obtained by extending KB with an axiom $D_0(a)$, and using an analogous argument to show that $B(a)$ is not entailed by any diversification of this knowledge base on a materialisation calculus of arity 2. □

Next, we consider a case where classification turns out to require a higher arity than instance retrieval.

Theorem 6. *Let $\mathcal{L}$ be a DL with GCIs, existential quantification, and nominal classes. Every materialisation calculus that is sound and complete for classification in $\mathcal{L}$ has arity three or more.*

Proof. The steps of the proof are similar to the proof of Theorem 5. Suppose that there is a binary materialisation calculus $K = \langle I, P, O\rangle$ such that I is defined for all axioms of $\mathcal{L}$, and O is defined for all axioms of the form $A \sqsubseteq B$. Let l be the maximal number of body atoms in any rule of P, and consider some $k > 2(l + 1)$. A knowledge base KB is defined to contain the following axioms:

- $D_i \sqsubseteq \exists S_i.D_{i+1}$ for all $i \in \{0, \dots, k\}$,
- $D_{k+1} \sqsubseteq \{a\}$,
- $\exists S_i.B_{i+1} \sqsubseteq B_i$ for all $i \in \{0, \dots, k\}$,
- $D_0 \sqsubseteq \exists R.A$, $A \sqsubseteq \{a\}$, $A \sqsubseteq B_{k+1}$.

Then KB entails $D_0 \sqsubseteq B_0$. Thus there is a proof tree T for deriving $O(D_0 \sqsubseteq B_0)$ for the program $I(\text{KB}) \cup I(\mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R}) \cup P$. Let $T' = \langle N, E, \lambda'\rangle$ be the diversified proof tree obtained from T by using renamed symbols s^n as in Definition 4, and let KB$'$ be the according diversified knowledge base.

We claim that KB$'$ has a model $\mathcal{I}$ that does not satisfy $D_0 \sqsubseteq B_0$. The definition of such a model is not difficult but somewhat unwieldy due to the effects of nominals, which may force some individuals to be identified in a model. We account for this by first defining $\approx$ to be the least equivalence relation on the set $\bigcup_{n\in N}\{a^n, A^n, D^n_{k+1}\}$ for which we find:

- if $A^n \sqsubseteq \{a^n\} \in \text{KB}'$, then $A^n \approx a^n$,
- if $D^n_{k+1} \sqsubseteq \{a^n\} \in \text{KB}'$, then $D^n_{k+1} \approx a^n$.

We use $[d]$ for denoting $\approx$ equivalence classes $[d] := \{e \mid e \approx d\}$. We now define a model $\mathcal{I}$ of KB$'$ over the domain $\Delta^{\mathcal{I}} := \bigcup_{n\in N}(\{[a^n], [A^n], [D^n_{k+1}]\} \cup \{D^n_i \mid 0 \le i \le k\})$. The interpretation is defined as follows:

- $(a^n)^{\mathcal{I}} := [a^n]$,
- $(D^n_0)^{\mathcal{I}} := \{D_0\}$ if $D^n_0 = D_0$; and $(D^n_0)^{\mathcal{I}} := \emptyset$ otherwise,
- $(D^n_i)^{\mathcal{I}} := \{D^n_i\}$ for $i = 1, \dots, k$, and $(D^n_{k+1})^{\mathcal{I}} := \{[D^n_{k+1}]\}$,
- $(S^n_i)^{\mathcal{I}} := \{\langle d_i, d'_{i+1}\rangle \mid d_i \sqsubseteq \exists S^n_i.d_{i+1} \in \text{KB}' \text{ and } d'_{i+1} \in \{d_{i+1}, [d_{i+1}]\} \cap \Delta^{\mathcal{I}}\}$,
- $(A^n)^{\mathcal{I}} := \{[A^n]\}$ if there is $m \in N$ such that $A^n = A^m$, $(D^m_0)^{\mathcal{I}} \neq \emptyset$, and $D^m_0 \sqsubseteq \exists R^m.A^m \in \text{KB}'$; and $(A^n)^{\mathcal{I}} := \emptyset$ otherwise,

– $(R^n)^{\mathcal{I}} := \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$,
– $(B^n_{k+1})^{\mathcal{I}} := \{[A^m] \mid (A^m)^{\mathcal{I}} \neq \emptyset \text{ and } A^m \sqsubseteq B^n \in \mathrm{KB}'\}$,
– for $i = k+1, \ldots, 1$ recursively define $(B^n_{i-1})^{\mathcal{I}} := \bigcup_{\exists S^m_i.B^m_i \sqsubseteq B^n_{i-1} \in \mathrm{KB}'} (\exists S^m_i.B^m_i)^{\mathcal{I}}$.

It is not hard to check that $\mathcal{I}$ is indeed a model of KB′. It remains to show that $\mathcal{I}$ does not entail $D_0 \sqsubseteq B_0$. To this end, note that $D_0 \in B_0^{\mathcal{I}}$ holds only if KB′ contains the following axioms:

– chains of axioms $d_0 \sqsubseteq s_0.d_1, \ldots, d_k \sqsubseteq s_k.d_{k+1}$ and $\exists s_0.b_1 \sqsubseteq b_0, \ldots, \exists s_k.b_{k+1} \sqsubseteq b_k$ where we find $d_0 = D_0$, $d_i = D^o_i$ for some $o \in N$, $s_i = S^o_i$ for some $o \in N$, $b_0 = B_0$, and $b_i = B^o_i$ for some $o \in N$,
– $D_0 \sqsubseteq \exists R'.A'$, $A' \sqsubseteq \{a'\}$, $A' \sqsubseteq B$, $d_{k+1} \sqsubseteq \{a'\}$.

For a contradiction, suppose that KB′ contains a set KB″ of axioms of the above form. As in the proof of Theorem 5, we find a node $o \in N$ such that Γ_o has three axioms of the form $d_i \sqsubseteq \exists s_i.d_{i+1}$ below it, and such that three other axioms of this form are not below it. The dependency graph for KB″ (see Fig. 6) is very similar to the one constructed for the proof of Theorem 5, and the arguments used for showing that Γ_o cannot exist due to the restrictions on the size of the interface of o are analogous to the ones given in the earlier proof.

As before, we conclude that KB″ cannot be contained in KB′, so $\mathcal{I}$ does not satisfy $D_0 \sqsubseteq B_0$, and thus the latter is not a consequence of KB′. Since T' is a proof tree for $I(\mathrm{KB}') \cup I(\mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R}) \cup P$ by Lemma 4, K derives $D_0 \sqsubseteq B_0$. So K cannot be sound, contradicting our assumption of its existence. □

It is interesting to note that the previous result does not extend to instance retrieval. The proof indicates why this is the case. The cyclic path that was required to exist in the dependency graph involves the axiom $D_0 \sqsubseteq \exists R'.A'$ which ensures non-emptiness of A' whenever D_0 is non-empty. If this axiom was present in another form, say $D^n_0 \sqsubseteq \exists R'.A'$ with $D^n_0 \neq D_0$, the result could not be concluded. The according instance retrieval problem, in contrast, includes an input axiom $D_0(c)$ from which the conclusion $B_0(c)$ must follow. In this case, the cyclic dependency is not needed: any class D^n_0 must be non-empty if the axiom $D^n_0(c^n)$ is present, and we do not need $D^n_0 = D_0$ to obtain the result. This explains on an intuitive level why we cannot extend Theorem 6 to instance retrieval like Theorem 5. To prove that a materialisation calculus of arity two would really suffice in this case, an according calculus would need to be specified – this is easy to do by eliminating the 4-ary `triple_sc` predicate from K_{sc} using the same methods as in the case of K_{sc^-} in Section 4.

Now we are ready to show that the arity of the materialisation calculus of Theorem 2 is optimal.

Theorem 7. *Let $\mathcal{L}$ be a DL with GCIs, existential quantification, role chains, and nominal classes. Every materialisation calculus that is sound and complete for classification in $\mathcal{L}$ has arity four or more.*

Proof. The steps of the proof are again similar to the proofs of Theorem 5 and 6. Suppose that there is a ternary materialisation calculus $K = \langle I, P, O \rangle$ such that I is defined on all axioms of $\mathcal{L}$, and O is defined for all axioms of the form $A \sqsubseteq B$. Let l be the

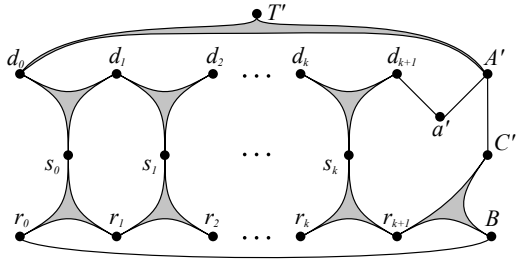

Fig. 7. Dependency graph for the proof of Theorem 7

maximal number of body atoms in any rule of P, and consider some $k > 3(l+1)$. A knowledge base KB is defined to contain the following axioms:

- $D_i \sqsubseteq \exists S_i.D_{i+1}$ for all $i \in \{0,\dots,k\}$,
- $D_0 \sqsubseteq \exists T.A$, $A \sqsubseteq \{a\}$, $D_{k+1} \sqsubseteq \{a\}$, $A \sqsubseteq C$,
- $C \sqsubseteq \exists R_{k+1}.B$,
- $S_i \circ R_{i+1} \sqsubseteq R_i$ for all $i \in \{0,\dots,k\}$,
- $\exists R_0.B \sqsubseteq B$.

Then KB entails $D_0 \sqsubseteq B$. Thus there is a proof tree T for deriving $O(D_0 \sqsubseteq B)$ for the program $I(\mathrm{KB}) \cup I(\mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R}) \cup P$. Let $T' = \langle N, E, \lambda' \rangle$ be the diversified proof tree obtained from T by using renamed symbols s^n as in Definition 4, and let KB′ be the according diversified knowledge base.

Combining the techniques of Theorem 5 and 6, it is not hard to construct a model $\mathcal{I}$ of KB′ in such a way that $\mathcal{I} \models D_0 \sqsubseteq B$ holds only if KB′ contains the following axioms:

- $D_0 \sqsubseteq T'.A'$,
- $A' \sqsubseteq \{a'\}$,
- $A' \sqsubseteq C'$,
- chains of axioms $d_0 \sqsubseteq s_0.d_1, \dots, d_k \sqsubseteq s_k.d_{k+1}$ and $s_0 \circ r_1 \sqsubseteq r_0, \dots, s_k \circ r_{k+1} \sqsubseteq r_k$ where we find $d_0 = D_0$, $d_i = D_i^o$ for some $o \in N$, $s_i = S_i^o$ for some $o \in N$, and $r_i = R_i^o$ for some $o \in N$,
- $C' \sqsubseteq r_{k+1}.B$,
- $d_{k+1} \sqsubseteq \{a'\}$,
- $\exists r_0.B \sqsubseteq B$.

Constructing the according model $\mathcal{I}$ is not difficult along the lines of the earlier proofs, and we omit the details here. For a contradiction, suppose that KB′ contains a set KB″ of axioms of the above form, the dependency graph of which is depicted in Fig. 7. By Lemma 5 and our choice of k, we can again find a node $o \in N$ such that Γ_o has four axioms of the form $d_i \sqsubseteq \exists s_i.d_{i+1}$ below it, and such that four other axioms of this form are not below it.

The proof that Γ_o cannot exist is very similar to the argument used in Theorem 5. Since K has arity 3, the maximal amount of symbols in the interface of o is the set of symbols used in three axioms of KB″. To show that this suffices, we note that the dependency graph of Fig. 7 now features two cyclic chains with nodes $d_0, d_1, \dots, d_{k+1}, a', A', d_0$

and $r_0, r_1, \ldots, r_{k+1}, B, r_0$. As in the proof of Theorem 5, we can argue that either of these chains has some (two or more) but not all (not even all but one) of its axioms below Γ_o. Thus two end nodes for the fragments of either chain must belong to the interface of o. Clearly, no two of those four nodes occur together in a single axiom, so the interface would have to be larger than possible for a ternary materialisation calculus.

Summing up Γ_o cannot exist, and thus KB$''$ cannot be contained in KB$'$. So $\mathcal{I}$ does not satisfy $D_0 \sqsubseteq B$, and thus the latter is not a consequence of KB$'$. Since T' is a proof tree for $I(\text{KB}') \cup I(\mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R}) \cup P$ by Lemma 4, K derives $D_0 \sqsubseteq B$. So K cannot be sound, contradicting our assumption of its existence. □

Just like Theorem 6, the previous result again applies to classification only. Indeed, Theorem 1 shows that a ternary instance retrieval calculus exists for a DL that includes existentials, nominals, and role chains.

Theorem 7 may be surprising, given that the calculus proposed in [2] for $\mathcal{EL}^{++}$ would be ternary in our notation. The explanation is that this algorithm is incomplete for classification; the proof of Theorem 7 can be used to find a suitable counter example. However, we do not need to use quite as many axioms as for the general case considered in the theorem:

- $D_0 \sqsubseteq \exists S.D_1$, $D_0 \sqsubseteq \exists T.A$,
- $A \sqsubseteq \{a\}$, $D_1 \sqsubseteq \{a\}$,
- $\exists S.A \sqsubseteq A$.

This knowledge base entails $D_0 \sqsubseteq A$ but the algorithm in [2] does not derive this. The reason is that the algorithm's derivation rule (CR6) is not able to infer $D_1 \sqsubseteq A$ since this is not universally true. Indeed, one can only infer "$D_1 \sqsubseteq A$ whenever D_0 is non-empty" but this statement cannot be represented by the algorithm in [2]. The algorithm features an auxiliary relation $E \rightsquigarrow_R F$ that signifies that "F must be non-empty whenever E is" but in our example, we only get $D_0 \rightsquigarrow_R A$ and $D_0 \rightsquigarrow_R D_1$ but not $D_1 \rightsquigarrow_R A$ as would be required for applying (CR6). Theorem 7 shows that this shortcoming cannot be overcome by introducing further special cases (e.g., one could allow (CR6) to apply in a situation with three classes as in our counter example): we can always find a larger counter example that is not covered. The only solution is to "contextualise" all derivations by conditions of the form "if D_0 is non-empty ..." which corresponds to an increase of the arity of derived statements by one. Note that the problem does not occur when restricting to instance retrieval (i.e. subsumption checking where the subsumed class is a nominal) since the auxiliary relation $\rightsquigarrow_R$ always takes all nominal classes into account. In our example, if D_0 would be replaced by $\{d_0\}$, then the relation $E \rightsquigarrow_R A$ is derived for any class E, especially for $E = D_1$, so $\{d_0\} \sqsubseteq A$ is inferred.

6 Summary and Conclusions

The focus of this work has been the study of inferencing calculi for $\mathcal{SROEL}(\sqcap, \times)$ and its fragments, and especially this paper is – to the best of our knowledge – the first to present a sound and complete polynomial time calculus for inferencing in a DL that is

so closely related to the OWL EL ontology language. For investigating properties of such calculi, we presented a simple framework for expressing materialisation calculi in terms of datalog. This revealed the arity of IDB predicates as an interesting measure for the worst-case space requirements of materialisation-based algorithms. While $\mathcal{SROEL}(\sqcap, \times)$ fragments without role chains and nominals admit classification calculi based on binary IDB predicates, the inclusion of either feature increases the required arity by one. Having both features, $\mathcal{SROEL}(\sqcap, \times)$ thus does not admit any sound and complete classification calculus of arity below four.

We are thus able to differentiate various $\mathcal{SROEL}(\sqcap, \times)$ fragments and inferencing tasks based on a measure that relates to the efficiency of actual implementations. Indeed, our findings agree with practical experiences that especially nominals and role chains are harder to implement efficiently than basic $\mathcal{EL}$ features.[5] Computational complexity has not been able to provide an explanation for such discrepancies, since all reasoning problems we consider are P-complete. In addition, our study also shows that various other features are not harder to implement than some of the most basic ones, thus providing guidance for deciding which features to implement or to use in an application.

Although there are standard implementation strategies for datalog reasoning, our study is independent of actual algorithms. A promising next step thus is to develop control strategies for implementing our calculi in a "pay-as-you-go" algorithm that minimises the potential negative impact of the occurrence of certain features. Moreover, we conjecture that our results about datalog arity can be further strengthened to obtain more direct statements about space complexity of almost arbitrary monotone calculi.

Acknowledgements The author thanks Yevgeny Kazakov for his valuable input, and the anonymous reviewers for helpful comments. This work was supported by DFG in project *ExpresST* and by EPSRC in project *ConDOR* (EP/G02085X/1).

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley (1994)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Kaelbling, L., Saffiotti, A. (eds.) Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05). pp. 364–369. Professional Book Center (2005)
3. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope further. In: Clark, K.G., Patel-Schneider, P.F. (eds.) Proc. OWLED 2008 DC Workshop on OWL: Experiences and Directions. CEUR Workshop Proceedings, vol. 496. CEUR-WS.org (2008)
4. 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)
5. Delaitre, V., Kazakov, Y.: Classifying $\mathcal{ELH}$ ontologies in SQL databases. In: Patel-Schneider, P.F., Hoekstra, R. (eds.) Proc. OWLED 2009 Workshop on OWL: Experiences and Directions. CEUR Workshop Proceedings, vol. 529. CEUR-WS.org (2009)
6. Hayes, P. (ed.): RDF Semantics. W3C Recommendation (10 February 2004), available at `http://www.w3.org/TR/rdf-mt/`

[5] Based on the author's experience implementing Orel [9], and personal communication with developers of DB [5] and CEL (`http://lat.inf.tu-dresden.de/systems/cel/`).

7. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
8. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06). pp. 57–67. AAAI Press (2006)
9. Krötzsch, M., Mehdi, A., Rudolph, S.: Orel: Database-driven reasoning for OWL 2 profiles. In: Haarslev, V., Toman, D., Weddell, G. (eds.) Proc. 23rd Int. Workshop on Description Logics (DL'10) (2010)
10. Krötzsch, M., Rudolph, S., Hitzler, P.: ELP: Tractable rules for OWL 2. In: Sheth et al. [14], pp. 649–664
11. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Language: Profiles. W3C Recommendation (27 October 2009), available at `http://www.w3.org/TR/owl2-profiles/`
12. Motik, B., Sattler, U.: A comparison of reasoning techniques for querying large description logic ABoxes. In: Hermann, M., Voronkov, A. (eds.) Proc. 13th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'01). LNCS, vol. 4246, pp. 227–241. Springer (2006)
13. Rudolph, S., Krötzsch, M., Hitzler, P.: Description logic reasoning with decision diagrams: Compiling $\mathcal{SHIQ}$ to disjunctive datalog. In: Sheth et al. [14], pp. 435–450
14. Sheth, A., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.): Proc. 7th Int. Semantic Web Conf. (ISWC'08), LNCS, vol. 5318. Springer (2008)