A Practical Acyclicity Notion for Query Answering over *Horn-$\mathcal{SRIQ}$* Ontologies

David Carral[1], Cristina Feier[2], and Pascal Hitzler[1]

[1] DaSe Lab, Wright State University, Dayton US
[2] Universität Bremen, Bremen Germany

Abstract. Conjunctive query answering over expressive Horn Description Logic ontologies is a relevant and challenging problem which, in some cases, can be addressed by application of the chase algorithm. In this paper, we define a novel acyclicity notion which provides a sufficient condition for termination of the restricted chase over *Horn-$\mathcal{SRIQ}$* TBoxes. We show that this notion generalizes most of the existing acyclicity conditions (both theoretically and empirically). Furthermore, this new acyclicity notion gives rise to a very efficient reasoning procedure. We provide evidence for this by providing a materialization based reasoner for acyclic ontologies which outperforms other state-of-the-art systems.

1 Introduction

Conjunctive query (CQ) answering over expressive Description Logic (DL) ontologies is a key reasoning task which remains unsolved for many practical purposes. Indeed, answering CQs over DL ontologies is quite intricate and often of high computational complexity [4, 8, 16]. Nevertheless, CQ answering over a major class of DLs, the so-called *Horn DLs*, can in some cases be addressed via application of the *chase algorithm*, a technique where all relevant consequences of an ontology are precomputed, allowing queries to be directly evaluated over the materialized set of facts. However, the chase is not guaranteed to terminate for all ontologies, and checking whether it does is not a straightforward procedure. It is thus an ongoing research endeavor to establish so-called *acyclicity conditions*; i.e., sufficient conditions which ensure termination of the chase.

The main contribution of this paper is the definition of *restricted chase acyclicity* (RCA_n), a novel acyclicity condition for *Horn-$\mathcal{SRIQ}$* ontologies (the DL *Horn-$\mathcal{SRIQ}$* may be informally described as the logic underpinning the deterministic fragment of OWL DL [9] minus nominals). If an ontology is proven to be RCA_n, then n-cyclic terms do not occur during the computation of the chase of such ontology and thus the chase is guaranteed to terminate.

In contrast with existing acyclicity notions [6] which deal with termination of the unrestricted, i.e. oblivious, chase of arbitrary sets of existential rules, we restrict our attention to the language *Horn-$\mathcal{SRIQ}$* and seek to achieve termination of the restricted chase algorithm [3]; this is a special variant of the standard chase in which the inclusion of further terms to satisfy existential restrictions is

avoided if such restrictions are already satisfied, and equality is dealt with via renaming. By considering such a chase algorithm we are able to devise acyclicity conditions which are more general than any other of the notions previously described.

On the theoretical side, we show that RCA_n is more general than *model-faithful acyclicity* (MFA) provided n is sufficiently large (linear in the size of ontology). As shown in [6], this is one of the most general acyclicity conditions for ontologies described to date, as it encompasses many other existing notions such as *joint acyclicity* [12], *super-weak acyclicity* [14] or the hybrid acyclicity notions presented in [2]. Furthermore, we show that deciding RCA_n membership is not harder than deciding MFA membership.

On the practical side, we empirically show that (i) RCA_n characterizes more real-world ontologies as acyclic than MFA. Furthermore, we demonstrate that (ii) the specific type of acyclicity captured by RCA_n results in a more efficient reasoning procedure. This is because acyclicity is still preserved in the case when employing renaming techniques when reasoning in the presence of equality. Thus, the use of cumbersome axiomatizations of equality such as *singularization* [14] can be avoided. Moreover, we report on an implementation of the restricted chase algorithm based on the datalog engine RDFOx [15] and show that (iii) it vastly outperforms state-of-the-art DL reasoners. To verify (i-iii), we complete an extensive evaluation with very encouraging results.

The rest of the paper is structured as follows: We start with some preliminaries in Section 2. Section 3 formally introduces the notions of oblivious and restricted chase, followed by an overview of MFA in Section 4. In Section 5 we introduce our new acyclicity notion RCA_n. Finally, Section 6 and Section 7 describe the evaluation of our work and list our conclusions, respectively.

An extended technical report for this paper with all the proofs and further information concerning the evaluation can be found at `http://dase.cs.wright.edu/publications/acyclicity-notion-cqa-over-horn-sriq-ontologies`.

2 Preliminaries

Rules We use the standard notions of *constants*, *function symbols* and *predicates*, where $\approx$ is the equality predicate, $\top$ is universal truth, and $\bot$ is universal falsehood. *Variables*, *terms*, *atoms* and *substitutions* are defined as usual. A *fact* is a ground atom; i.e., an atom without occurrences of variables. As customary, every term t is associated with some *depth* $dep(t) \geq 0$. Furthermore, we often abbreviate a vector of terms $t_1, \ldots, t_n$ as $\boldsymbol{t}$ and identify $\boldsymbol{t}$ with the set $\{t_1, \ldots, t_n\}$. In a similar manner, we often identify a conjunction of atoms $\phi_1 \wedge \ldots \wedge \phi_n$ with the set $\{\phi_1, \ldots, \phi_n\}$. With $\phi(\boldsymbol{x})$ we stress that $\boldsymbol{x} = x_1, \ldots, x_n$ are the free variables occurring in the formula ϕ.

Let t be some ground term and c some constant. Let t_c be the term obtained from t by replacing every occurrence of a constant by c, i.e., $f(d, g(e))_c = f(c, g(c))$. The notation is analogously extended to facts and sets of facts.

A term t' is a *subterm* of another term t if and only if $t' = t$, or $t = f(\boldsymbol{s})$ and t' is a subterm of some $s \in \boldsymbol{s}$; if additionally $t' \neq t$, then t' is a *proper subterm* of t. A term t is *n-cyclic* if and only if there exists a sequence of terms of the form $f(\boldsymbol{s_1}), \ldots, f(\boldsymbol{s_{n+1}})$ such that $f(\boldsymbol{s_{n+1}})$ is a subterm of t and, for every $i = 1, \ldots, n$, $f(\boldsymbol{s_i})$ is a proper subterm of $f(\boldsymbol{s_{i+1}})$. We simply refer to 1-cyclic terms as *cyclic*.

A *rule* is a first-order logic (FOL) formula of one of the forms

$$\forall \boldsymbol{x} \forall \boldsymbol{z} [\beta(\boldsymbol{x}, \boldsymbol{z}) \rightarrow \exists \boldsymbol{y} \eta(\boldsymbol{x}, \boldsymbol{y})] \qquad \text{or} \tag{1}$$

$$\forall \boldsymbol{x} [\beta(\boldsymbol{x}) \rightarrow x \approx y], \tag{2}$$

where β and η are non-empty conjunctions of atoms which do not contain occurrences of constants, function symbols nor of the predicate $\approx$; $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ are pairwise disjoint; and $x, y \in \boldsymbol{x}$. To simplify the notation, we frequently omit the universal quantifiers from rules. As customary, we refer to rules of the forms (1) and (2) as *tuple generating dependencies* (*TGDs*) and *equality generating dependencies* (*EGDs*), respectively.

Given a set of rules $\mathcal{R}$, we define $\mathcal{R}^{\exists}$ and $\mathcal{R}^{\forall}$ as the sets of all the TGDs in $\mathcal{R}$ which do and do not contain existentially quantified variables, respectively. Moreover, let $\mathcal{R}^{\approx}$ be the set of all EGDs in $\mathcal{R}$. A *program* is a tuple $\langle \mathcal{R}, \mathcal{I} \rangle$ where $\mathcal{R}$ is a set of rules and $\mathcal{I}$ is an *instance*; i.e., a finite set of equality-free facts.

The main reasoning task we are investigating in this paper is CQ answering. Nevertheless, for the rest of the paper, we restrict our attention to the simpler task of CQ entailment of *boolean conjunctive queries* (BCQs). This is without loss of generality since CQ answering can be reduced to checking entailment of BCQs. A *BCQ*, or simply a *query*, is a formula of the form $\exists \boldsymbol{y} \eta(\boldsymbol{y})$ where η is a conjunction of atoms not containing occurrences of constants, function symbols nor $\approx$.

For the remainder of the paper, we assume that $\top$ and $\bot$ are treated as ordinary unary predicates and that the semantics of $\top$ is captured explicitly in any program $\mathcal{P} = \langle \mathcal{R}, \mathcal{I} \rangle$ by including the rule $p(x_1, \ldots, x_n) \rightarrow \top(x_1) \wedge \ldots \wedge \top(x_n)$ in $\mathcal{R}$ for every predicate p with arity n occurring in $\mathcal{P}$.

We interpret programs under standard FOL semantics with true equality. As usual, a program $\mathcal{P}$ is *satisfiable* if and only if $\mathcal{P} \not\models \exists y \bot(y)$. Furthermore, given some query γ, we write $\mathcal{P} \models \gamma$ to indicate that $\mathcal{P}$ *entails* γ.

We will later employ skolemization to define the consequences of a TGD over a set of facts. The *skolemization* $sk(\rho)$ of some TGD $\rho = \beta(\boldsymbol{x}, \boldsymbol{z}) \rightarrow \exists \boldsymbol{y} \eta(\boldsymbol{x}, \boldsymbol{y})$ is the rule $\beta(\boldsymbol{x}, \boldsymbol{z}) \rightarrow \eta(\boldsymbol{x}, \boldsymbol{y}) \sigma_{sk}$ where σ_{sk} is a substitution mapping every $y \in \boldsymbol{y}$ into $f^y_\rho(\boldsymbol{x})$ where f^y_ρ is a fresh function unique for every variable y and TGD ρ.

Description Logics We next define the syntax and semantics of the ontology language *Horn-$\mathcal{SRIQ}$* [13]. We assume basic familiarity with DL, and refer the reader to the literature for further details [1]. Without loss of generality, we restrict our attention to ontologies in a normal form close to the one from [13].

A *DL signature* is a tuple $\langle N_C, N_R, N_I \rangle$ where N_C, N_R and N_I are infinite countable and mutually disjoint sets of *concept names*, *role names* and *indi-*

$$
\begin{array}{rcl}
A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B & \mapsto & A_1(x) \wedge \ldots \wedge A_n(x) \rightarrow B(x) \\
A \sqsubseteq \forall R.B & \mapsto & A(x) \wedge R(x,y) \rightarrow B(y) \\
A \sqsubseteq \leq 1\, R.B & \mapsto & A(x) \wedge R(x,y) \wedge B(y) \wedge R(x,z) \wedge B(z) \rightarrow y \approx z \\
A \sqsubseteq \exists R.B & \mapsto & A(x) \rightarrow \exists y[R(x,y) \wedge B(y)] \\
S \sqsubseteq R & \mapsto & S(x,y) \rightarrow R(x,y) \\
S^- \sqsubseteq R & \mapsto & S(y,x) \rightarrow R(x,y) \\
S \circ V \sqsubseteq R & \mapsto & S(x,y) \wedge V(y,z) \rightarrow R(x,z)
\end{array}
$$

Fig. 1. Mapping axioms α to rules $\Pi(\alpha)$, where $A_{(i)}, B \in N_C$, $R, S, V \in N_R$.

viduals, respectively, such that $\{\bot, \top\} \subseteq N_C$. A *role* is an element of $N_R^- = N_R \cup \{R^- \mid R \in N_R\}$. A *TBox axiom* is a formula of one of the forms given on the left hand side of the mappings in Figure 1. TBox axioms of the form $A \sqsubseteq \exists R.B$ are also referred as *existential axioms*. An *ABox axiom* is a formula of the form $A(a)$ or $R(a,b)$ where $A \in N_C$, $R \in N_R$ and $a, b \in N_I$. An *axiom* is either a TBox or an ABox axiom. As usual, we simply refer to a set of TBox (resp. ABox) axioms as a *TBox* (resp. an *ABox*).

A *Horn-$\mathcal{SRIQ}$ ontology* $\mathcal{O}$ (or simply an *ontology*) is some tuple $\langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ and $\mathcal{A}$ are a TBox and an ABox, respectively, which satisfies the usual conditions [10].

Due to the close correspondence between ontologies and programs, we define the semantics of the former by means of a mapping into the latter. Given some TBox $\mathcal{T}$, let $\mathcal{R}_\mathcal{T} = \Pi(\mathcal{T})$. Given some ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, let $\mathcal{P}(\mathcal{O}) = (\mathcal{R}_\mathcal{T}, \mathcal{A})$ where Π is the function from Figure 1. We say that $\mathcal{O}$ is *satisfiable* if and only if the program $\mathcal{P}(\mathcal{O})$ is satisfiable. Furthermore, $\mathcal{O}$ *entails* a query γ, written $\mathcal{O} \models \gamma$, if and only if $\mathcal{P}(\mathcal{O})$ is unsatisfiable or $\mathcal{P}(\mathcal{O})$ entails γ.

3 The Chase Algorithm

In this section we present two variants of the chase algorithm, which are somewhat similar to the oblivious and restricted chase from [3], and elaborate about how such procedures may be used to solve CQ entailment over ontologies.

Definition 1. *A fact ϕ is an* oblivious consequence *of a TGD $\rho = \beta(\boldsymbol{x}, \boldsymbol{z}) \rightarrow \exists \boldsymbol{y} \eta(\boldsymbol{x}, \boldsymbol{y})$ on a set of facts $\mathcal{F}$ if and only if there is some substitution σ with $\beta(\boldsymbol{x}, \boldsymbol{z})\sigma \subseteq \mathcal{F}$ and $\phi \in sk(\eta(\boldsymbol{x}, \boldsymbol{y}))\sigma$ where $sk(\eta(\boldsymbol{x}, \boldsymbol{y}))$ is the head of the (skolemized) TGD $sk(\rho)$. A fact ϕ is a* restricted consequence *of ρ on $\mathcal{F}$ if and only if there is a substitution σ with (1) $\beta(\boldsymbol{x}, \boldsymbol{z})\sigma \subseteq \mathcal{F}$ and $\phi \in sk(\eta(\boldsymbol{x}, \boldsymbol{y}))\sigma$, and (2) there is no substitution $\tau \supseteq \sigma$ with $\eta(\boldsymbol{x}, \boldsymbol{y})\tau \subseteq \mathcal{F}$.*

The result of obliviously applying *ρ to $\mathcal{F}$, written $\rho_O(\mathcal{F})$, is the set of all* oblivious consequences *of ρ on $\mathcal{F}$. The result of* obliviously applying *a set of*

TGDs $\mathcal{R}$ to $\mathcal{F}$, written $\mathcal{R}_O(\mathcal{F})$, is the set $\bigcup_{\rho\in\mathcal{R}} \rho_O(\mathcal{F}) \cup \mathcal{F}$. The result of restrictively applying *ρ to $\mathcal{F}$ (resp., $\mathcal{R}$ to $\mathcal{F}$), written $\rho_R(\mathcal{T})$ (resp., $\mathcal{R}_R(\mathcal{T})$), is analogously defined.*

Definition 2. *Let $\rightsquigarrow$ be some total strict order over the set of all terms such that $t \rightsquigarrow u$ only if $dep(t) \leq dep(u)$. Furthermore, we say that t is greater than u with respect to $\rightsquigarrow$ to indicate $t \rightsquigarrow u$.*

Given a set of EGDs $\mathcal{R}$ and a set of facts $\mathcal{F}$, let $\mapsto^{\mathcal{R}}_{\mathcal{F}}$ be the minimal congruence relation over terms such that $t \mapsto^{\mathcal{R}}_{\mathcal{F}} u$ if and only if there exists some $\beta(\boldsymbol{x}) \rightarrow x \approx y \in \mathcal{R}$ and some substitution σ with $\beta(\boldsymbol{x})\sigma \subseteq \mathcal{F}$, $\sigma(x) = t$ and $\sigma(y) = u$. Let $\mathcal{R}(\mathcal{F})$ be the set that is obtained from $\mathcal{F}$ by replacing all occurrences of every term t by u where u is the greatest term with respect to $\rightsquigarrow$ such that $t \mapsto^{\mathcal{R}}_{\mathcal{F}} u$.

Note that we define consequences with respect to sets of rules instead of simply (single) rules as it is customary [3]. This allows us to define the chase as a deterministic procedure (modulo $\rightsquigarrow$). Also, unlike in [3], where a lexicographic order is used to direct the replacement of terms, we employ a type of order which ensures that terms are always replaced by terms of equal or lesser depth. This effectively precludes some "deeper" terms from being introduced during the computation of the chase.

Definition 3. *Let $\mathcal{P} = \langle \mathcal{R}, \mathcal{I} \rangle$ be some program. The* oblivious chase sequence *of $\mathcal{P}$ is the sequence $\mathcal{F}_0, \mathcal{F}_1, \ldots$ such that $\mathcal{F}_1 = \mathcal{I}$ and, for all $i \geq 1$, $\mathcal{F}_i$ is the set of facts defined as follows.*

- *If $\mathcal{R}^{\approx}(\mathcal{F}_{i-1}) \neq \mathcal{F}_{i-1}$, then $\mathcal{F}_i = \mathcal{R}^{\approx}(\mathcal{F}_{i-1})$.*
- *If $\mathcal{F}_{i-1} = \mathcal{R}^{\approx}(\mathcal{F}_{i-1})$ and $\mathcal{F}_{i-1} \neq \mathcal{R}^{\forall}_O(\mathcal{F}_{i-1})$, then $\mathcal{F}_i = \mathcal{R}^{\forall}_O(\mathcal{F}_{i-1})$.*
- *Otherwise, $\mathcal{F}_i = \mathcal{R}^{\exists}_O(\mathcal{F}_{i-1})$.*

The restricted chase sequence *of $\mathcal{P}$ is defined analogously.*

For the sake of brevity, we frequently denote the oblivious (resp., restricted) chase sequence of a program $\mathcal{P}$ with $\mathcal{P}^1_O, \mathcal{P}^2_O, \ldots$ (resp., $\mathcal{P}^1_R, \mathcal{P}^2_R, \ldots$)

Definition 4. *Let $\mathcal{P}$ be some program and let $\mathcal{R}$ be some set of rules. Then, the* oblivious chase *of $\mathcal{P}$ is the set $OC(\mathcal{P}) = \bigcup_{i\in\mathbb{N}} \mathcal{P}^i_O$. The* restricted chase *of $\mathcal{P}$, written $RC(\mathcal{P})$, is defined analogously.*

The oblivious (resp., restricted) chase of $\mathcal{P}$ terminates *if and only if there is some i such that, for all $j \geq i$, $\mathcal{P}^i_O = \mathcal{P}^j_O$. Furthermore, the oblivious (resp., restricted) chase of a set of rules $\mathcal{R}$* terminates *if the oblivious (resp., restricted) chase of every program of the form $\langle \mathcal{R}, \mathcal{I} \rangle$ terminates.*

Our definition of the chase sequence ensures that rules which do not contain existentially quantified variables are always applied with a higher priority than rules that do. Note that, by postponing the application of rules with existential variables, we may prevent them from introducing further consequences.

The (restricted or oblivious) chase of a program can be employed to solve CQ entailment [3]. I.e., a program $\mathcal{P}$ entails a query γ, written $\mathcal{P} \models \gamma$, if and

$$
\begin{aligned}
\mathcal{T} &= \{\mathit{Film} \sqsubseteq \exists \mathit{isProdBy}.\mathit{Producer}, \mathit{Producer} \sqsubseteq \exists \mathit{prod}.\mathit{Film},\\
&\qquad \mathit{isProdBy}^{-} \sqsubseteq \mathit{prod}, \mathit{prod}^{-} \sqsubseteq \mathit{isProdBy}\}\\
\mathcal{O} &= \langle \mathcal{T}, \{\mathit{Film}(\mathit{AI})\}\rangle\\
\mathcal{R}_{\mathcal{T}} &= \{\rho = \mathit{Film}(x) \rightarrow \exists y[\mathit{isProdBy}(x,y) \wedge \mathit{Producer}(y)],\\
&\qquad \upsilon = \mathit{Producer}(x) \rightarrow \exists y[\mathit{prod}(x,y) \wedge \mathit{Film}(y)],\\
&\qquad \mathit{isProdBy}(y,x) \rightarrow \mathit{prod}(x,y), \mathit{prod}(y,x) \rightarrow \mathit{isProdBy}(x,y)\}\\
\mathcal{P}(\mathcal{O}) &= \langle \mathcal{R}_{\mathcal{T}}, \{\mathit{Film}(\mathit{AI})\}\rangle\\
\mathcal{P}(\mathcal{O})^{1}_{R} &= \{\mathit{Film}(\mathit{AI}), \mathit{isProdBy}(\mathit{AI}, f^{y}_{\rho}(\mathit{AI})), \mathit{Producer}(f^{y}_{\rho}(\mathit{AI}))\}\\
\mathcal{P}(\mathcal{O})^{2}_{R} &= \{\mathit{prod}(f^{y}_{\rho}(\mathit{AI}), \mathit{AI})\} \cup \mathcal{P}(\mathcal{O})^{1}_{O}\\
\mathit{RC}(\mathcal{P}(\mathcal{O})) &= \mathcal{P}(\mathcal{O})^{2}_{O}\\
\mathit{OC}(\mathcal{P}(\mathcal{O})) &= \mathit{RC}(\mathcal{P}(\mathcal{O})) \cup \{\mathit{prod}(f^{y}_{\rho}(\mathit{AI}), f^{y}_{\upsilon}(f^{y}_{\rho}(\mathit{AI}))), \mathit{Film}(f^{y}_{\upsilon}(f^{y}_{\rho}(\mathit{AI}))), \ldots\}
\end{aligned}
$$

Fig. 2. Ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A}\rangle$, program $\mathcal{P}(\mathcal{O})$ and the chase of $\mathcal{P}(\mathcal{O})$.

only if either $\mathit{OC}(\mathcal{P}) \models \exists y \bot(y)$ or $\mathit{OC}(\mathcal{P}) \models \gamma$ (resp., $\mathit{RC}(\mathcal{P}) \models \exists y \bot(y)$ or $\mathit{RC}(\mathcal{P}) \models \gamma$). Thus, we may also use the chase to solve CQ entailment over ontologies: An ontology $\mathcal{O}$ entails a query γ if and only if $\mathit{OC}(\mathcal{P}(\mathcal{O})) \models \exists y \bot(y)$ or $\mathit{OC}(\mathcal{P}(\mathcal{O})) \models \gamma$ (resp., $\mathit{RC}(\mathcal{P}(\mathcal{O})) \models \exists y \bot(y)$ or $\mathit{RC}(\mathcal{P}(\mathcal{O})) \models \gamma$).

For readability purposes, we say that the oblivious (resp. restricted) chase of some ontology $\mathcal{O}$ *terminates* if and only if the oblivious (resp. restricted) chase of $\mathcal{P}(\mathcal{O})$ terminates. The oblivious (resp. restricted) chase of some TBox $\mathcal{T}$ *terminates* if and only if if the oblivious (resp. restricted) chase of $\mathcal{R}_{\mathcal{T}}$ terminates.

As expected, the restricted chase has a better behavior than the oblivious chase; i.e., in some cases, the former might terminate when the latter does not:

Example 5. Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A}\rangle$ be as in Figure 2. The figure depicts also the computation of the oblivious chase and that of the restricted chase of $\mathcal{P}(\mathcal{O})$. In this case, $\mathit{RC}(\mathcal{P}(\mathcal{O}))$ terminates whereas $\mathit{OC}(\mathcal{P}(\mathcal{O}))$ does not.

4 Model Faithful Acyclicity

In this section we briefly describe Model Faithful Acyclicity (MFA) [6], one of the most general acyclicity conditions for sets of rules. MFA guarantees the termination of the oblivious chase of a program by imposing that no cyclic term occurs in the chase. Note that, a condition such as MFA can be applied to check whether a TBox $\mathcal{T}$ is acyclic; i.e., $\mathcal{T}$ is MFA if and only if $\mathcal{R}_{\mathcal{T}}$ is MFA.

When one is interested in checking the termination of the oblivious chase with respect to every possible instance, it is enough to check termination with respect to a special instance, the *critical instance* [14]. The critical instance is the minimal set which contains all possible atoms that can be formed using the relational symbols which occur in TGDs and the special constant $\star$. Such a strategy is used by MFA to guarantee termination of a set of rules.

While the actual definition of MFA does not preclude the existence of EGDs, equality is assumed to be axiomatized, and thus it is treated as a regular predicate (EGDs are de facto TGDs). To reflect such treatment we will use the special predicate Eq to denote equality. However, as the following example shows, the presence of equality in a set of TGDs frequently makes the MFA membership test fail.

Example 6. Let Σ be the following set of rules and let Σ' be the set of rules that result from axiomatizing the equality predicate as usual (see Section 2.1 of [6]). Furthermore, let $\mathcal{I}_\star(\Sigma')$ be the critical instance of Σ'.

$$\begin{aligned}
\Sigma &= \{A(x) \wedge B(x) \rightarrow \exists y[R(x,y) \wedge B(y)], R(z,x_1) \wedge R(z,x_2) \rightarrow Eq(x_1,x_2)\} \\
\mathsf{Eq} &= \{\top(x) \rightarrow Eq(x,x), Eq(x,y) \rightarrow Eq(y,x), Eq(x,z) \wedge Eq(z,y) \rightarrow Eq(x,y)\} \\
\Sigma' &= \{A(x) \wedge Eq(x,y) \rightarrow A(y), R(x,y) \wedge Eq(x,z) \rightarrow R(z,y), \\
&\qquad R(x,y) \wedge Eq(y,z) \rightarrow R(x,z)\} \cup \Sigma \cup \mathsf{Eq} \\
\mathcal{I}_\star(\Sigma') &= \{A(\star), R(\star,\star), Eq(\star,\star)\}
\end{aligned}$$

The oblivious chase of $(\Sigma', \mathcal{I}_\star(\Sigma'))$ does not terminate.

$$\begin{aligned}
(\Sigma', \mathcal{I}_\star(\Sigma'))^1_O &= \{R(\star, f(\star)), B(f(\star)), Eq(\star, f(\star))\} \cup \mathcal{I}_\star(\Sigma') \\
(\Sigma', \mathcal{I}_\star(\Sigma'))^2_O &= \{A(f(\star)), R(f(\star), f(f(\star))), B(f(f(\star))), \ldots\} \\
&\cdots\cdots\cdots\cdots\cdots\cdots
\end{aligned}$$

To avoid this situation, the use of *singularization* [14], a somewhat "less-harmful" axiomatization of equality, is proposed in [6].

Definition 7. *A* singularization of a rule ρ *is the rule* ρ' *that results from performing the following transformation for every variable* v *in the body of* ρ*:*

- *Rename each occurrence of* v *using different fresh variables* $v_1, \ldots, v_n$*,*
- *pick some* $j = 1, \ldots, n$ *and add the atoms* $Eq(v_1, v_j), \ldots, Eq(v_n, v_j)$ *to the body of* ρ *and*
- *replace any occurrence of* v *in the head of* ρ *with* v_j*.*

Let Σ *be a set of TGDs and let* Eq *be the set from Example 6. A* singularization *of* Σ *is a set of TGDs* Σ' *which contains* Eq *and exactly one singularization of every* $\rho \in \Sigma$*. Let* $Sing(\Sigma)$ *be the set of all possible singularizations of* Σ*.*

Example 8. Rule $A(x) \wedge B(x) \rightarrow \exists y[R(x,y) \wedge B(y)]$ from Example 6 admits two possible singularizations: (i) $A(x_1) \wedge B(x_2) \wedge Eq(x_2,x_1) \rightarrow \exists y[R(x_1,y) \wedge B(y)]$ and (ii) $A(x_1) \wedge B(x_2) \wedge Eq(x_1,x_2) \rightarrow \exists y[R(x_2,y) \wedge B(y)]$.

Note that, for any $\Sigma' \in Sing(\Sigma)$, if Σ' is MFA, then the oblivious chase of Σ' can be used to answer queries on Σ [6]. The use of singularization along with MFA gives rise to the following acyclicity notions.

Definition 9. *For a set of TGDs* Σ*, if there is some* $\Sigma' \in Sing(\Sigma)$ *which is MFA, then* Σ *is said to be* $MFA^\exists$*. If every* $\Sigma' \in Sing(\Sigma)$ *is MFA, then* Σ *is* $MFA^\forall$*.*

To some extent, the use of singularization solves the problems with equality: One can check that Σ in Example 6 is $\text{MFA}^{\exists}$, but not $\text{MFA}^{\forall}$. Nevertheless, due to the high number of possible singularizations, it is frequently not feasible to check $\text{MFA}^{\exists}$ or $\text{MFA}^{\forall}$ membership. A simpler alternative is to check whether $\bigcup_{\Sigma' \in Sing(\Sigma)} \Sigma'$ is MFA. If that is the case, then Σ is said to be $\text{MFA}^{\cup}$. Note that in the case of *Horn-$\mathcal{SRIQ}$* TBoxes, $|\bigcup_{\Sigma' \in Sing(\Sigma)} \Sigma'|$ is actually polynomial in $|\Sigma|$ and, as such, $\text{MFA}^{\cup}$ is more feasible to check. Thus, we will use $\text{MFA}^{\cup}$ as a baseline for the evaluation of the new acyclicity condition RCA_n, which is introduced in the next section.

5 Restricted Chase Acyclicity

While MFA is quite a general acyclicity condition, it has two main drawbacks:

1. It only considers the oblivious chase, which as we have seen in Example 5, might not terminate (even though the restricted chase does!), and
2. its treatment of equality via singularization is cumbersome and inefficient in practice. Not only $\text{MFA}^{\exists}$ and $\text{MFA}^{\forall}$ are difficult to check, but even after a set of TGDs are established to belong to some MFA subclass, one has to employ a singularized program for reasoning purposes.

In this section, we present RCA_n, an acyclicity notion with neither of these drawbacks: RCA_n verifies termination of the restricted chase of a TBox and does not require the use of cumbersome axiomatizations of the equality predicate. Furthermore, unlike MFA, RCA_n allows for the presence of cyclic terms in the chase up to a given depth n.

Since we are primarily interested in termination of the restricted chase of a *Horn-$\mathcal{SRIQ}$* TBox, one might wonder why we do not simply check for termination of the restricted chase for such a TBox with respect to the critical instance, as it is done in the previous section with the oblivious chase. Unfortunately, this is not possible: The restricted chase of any set of existential rules always terminates with respect to the critical instance. Thus, we have to devise more sophisticated techniques to check the termination of the restricted chase. We start by introducing the notion of an overchase for a TBox.

Definition 10. *A set of facts $\mathcal{V}$ is an* overchase *for some TBox $\mathcal{T}$ if and only if, for every $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, $RC(\mathcal{P}(\mathcal{O}))_\star \subseteq \mathcal{V}$.*

Given some TBox $\mathcal{T}$, an overchase for $\mathcal{T}$ may be intuitively regarded as an over-approximation of the restricted chase of $\mathcal{T}$.

Lemma 11. *If there exists a finite overchase for a TBox, then the restricted chase of such TBox terminates.*

Thus, to determine whether the chase of a TBox $\mathcal{T}$ terminates, we introduce a procedure to compute an overchase for $\mathcal{T}$ and a means to check its termination. We proceed with some preliminary notions and notation.

Definition 12. *Let $\mathcal{T}$ be some TBox and t a term. Let $\mathcal{I}(t)$ be the set of facts defined as follows: If t is of the form $f^y_\rho(s)$ where $\rho = A(x) \rightarrow \exists y[R(x,y) \wedge B(y)]$, then $\mathcal{I}(t) = \{A(s), R(s,t), B(t)\} \cup \mathcal{I}(s)$; otherwise, $\mathcal{I}(t) = \emptyset$. Furthermore, we introduce the program $\mathcal{U}(\mathcal{T},t) = \langle \mathcal{R}^{\forall}_{\mathcal{T}} \cup \mathcal{R}^{\approx}_{\mathcal{T}}, \mathcal{I}(t)\rangle$.*

Intuitively, the restricted chase of the program $\mathcal{U}(\mathcal{T},t)$ can be regarded as some kind of under-approximation of the facts that must occur in the chase of every program of the form $\mathcal{P}(\langle \mathcal{T}, \mathcal{A}\rangle)$ where t occurs. I.e., if t occurs in the restricted chase sequence of any program $\mathcal{P}(\langle \mathcal{T}, \mathcal{A}\rangle)$, then the facts in the restricted chase of $\mathcal{U}(\mathcal{T},t)$ must also occur (up to renaming) in the chase sequence of such program. Furthermore, due to the special priority of application of the rules during the computation of the chase, the facts in the restricted chase of $\mathcal{U}(\mathcal{T},t)$ must occur in the restricted chase sequence of every program of the form $\mathcal{P}(\langle \mathcal{T}, \mathcal{A}\rangle)$ before any successors of t are introduced.

Example 13. Let $\mathcal{O}$, ρ and υ be the ontology and rules from Example 5. Then, by Definition 12:

$$\mathcal{I}(f^y_\rho(AI)) = \{\mathit{Film}(AI), \mathit{isProdBy}(AI, f^y_\rho(AI)), \mathit{Producer}(f^y_\rho(AI))\} \text{ and}$$
$$RC(\mathcal{U}(\mathcal{T}, f^y_\rho(AI))) = \{\mathit{prod}(f^y_\rho(AI), AI)\} \cup \mathcal{I}(f^y_\rho(AI)).$$

All the facts in the restricted chase of $\mathcal{U}(\mathcal{T},t)$ occur in the restricted chase sequence of $\mathcal{P}(\mathcal{O})$ before any successors of term $f^y_\rho(AI)$ are introduced. This is because the rule $\mathit{isProdBy}(y,x) \rightarrow \mathit{prod}(x,y)$ is applied with a higher priority than the rule $\upsilon = \mathit{Producer}(x) \rightarrow \exists y[\mathit{prod}(x,y) \wedge \mathit{Film}(y)]$.

Given a TBox $\mathcal{T}$ and some term of the form $f^y_\rho(t)$, we can in some cases conclude that such a term may never occur during the computation of the restricted chase of every program of the form $\mathcal{P}(\langle \mathcal{T}, \mathcal{A}\rangle)$ by carefully inspecting the facts in the set $\mathcal{U}(\mathcal{T},t)$.

Definition 14. *Let $\mathcal{T}$ be a TBox and t a term of the form $f^y_\rho(s)$ where $\rho = A(x) \rightarrow \exists y[R(x,y) \wedge B(y)]$. We say that a term t is* restricted with respect to $\mathcal{T}$ *if and only if there is some term u with $\{R([s],u), B(u)\} \subseteq RC(\mathcal{U}(\mathcal{T},s))$ where $[s] = [v]$, if s is replaced by v during the computation of the restricted chase sequence; and $[s] = s$, otherwise.*

We often simply say that a term is "restricted", instead of "restricted with respect to $\mathcal{T}$," if the TBox $\mathcal{T}$ is clear from the context.

Lemma 15. *Let $\mathcal{T}$ be a TBox and t a restricted term. Then, for every possible $\mathcal{O} = \langle \mathcal{T}, \mathcal{A}\rangle$, $t \notin RC(\mathcal{P}(\mathcal{O}))$.*

Proof. (Sketch) Let t be a term of the form $f^y_\rho(s)$ where $\rho = A(x) \rightarrow \exists y(R(x,y) \wedge B(y))$. We can verify that, if t occurs during the computation of the chase sequence, then every fact $RC(\mathcal{U}(\mathcal{T},s))$ will also be included in such chase sequence before any new terms are introduced. Thus, if t is indeed restricted, there must be some u with $R([s],u)$ and $B(u)$ occurring in the chase sequence. Therefore, by the definition of the chase, the term t may never be derived.

$\forall$-rule	if	there is some TGD of the form $\rho = \beta(\boldsymbol{x}, \boldsymbol{y}) \rightarrow \eta(\boldsymbol{x}) \in \mathcal{R}_\mathcal{T}$
	then	$\mathcal{V}_\mathcal{T} \rightarrow \rho_R(\mathcal{V}_\mathcal{T}) \cup \mathcal{V}_\mathcal{T}$
$\exists$-rule	if	there is some TGD of the form $\rho = A(x) \rightarrow \exists y[R(x, y) \wedge B(y)] \in \mathcal{R}_\mathcal{T}$ and there exists some substitution σ such that (i) $A(x)\sigma \subseteq \mathcal{V}_\mathcal{T}$ and (ii) $f^y_\rho(x)\sigma$ is not restricted with respect to $\mathcal{T}$
	then	$\mathcal{V}_\mathcal{T} \rightarrow \{R(x, f^y_\rho(x)), B(f^y_\rho(x))\}\sigma \cup \mathcal{V}_\mathcal{T}$
$\approx$-rule	if	there is some EGD $\beta(\boldsymbol{x}, \boldsymbol{y}) \rightarrow x \approx y \in \mathcal{R}_\mathcal{T}$ and there exists some substitution σ such that $\beta(\boldsymbol{x}, \boldsymbol{y})\sigma \subseteq \mathcal{V}_\mathcal{T}$
	then	$\mathcal{V}_\mathcal{T} \rightarrow \{Eq(x, y), Eq(y, x)\}\sigma \cup \mathcal{V}_\mathcal{T}$
Eq-rule	if	there are some terms t, u and u_i where $i = 1, \ldots, n$ and some predicate p such that (i) $p \neq Eq$, (ii) $\{Eq(t, u), p(u_1, \ldots, u_n)\} \subseteq \mathcal{V}_\mathcal{T}$, (iii) $dep(t) \leq dep(u)$ and (iv) $u = u_j$ for some $j = 1, \ldots, n$
	then	$\mathcal{V}_\mathcal{T} \rightarrow \{p(u_1, \ldots, u_n)\}[u/t] \cup \mathcal{V}_\mathcal{T}$

Fig. 3. Expansion rules for the construction of $\mathcal{V}_\mathcal{T}$.

Example 16. Let $\mathcal{T}$, ρ and υ be the TBox and rules from Example 5. We proceed to show that the term $f^y_\rho(f^y_\upsilon(AI))$ is restricted. First, we compute the restricted chase of $\mathcal{U}(\mathcal{T}, f^y_\upsilon(AI))$.

$$RC(\mathcal{U}(\mathcal{T}, f^y_\upsilon(AI))) = \{Producer(AI), prod(AI, f^y_\upsilon(AI)), \\ Film(f^y_\upsilon(AI)), isProdBy(f^y_\upsilon(AI), AI)\}$$

Note that $\{isProdBy(f^y_\upsilon(AI), AI), Producer(AI)\} \subseteq RC(\mathcal{U}(\mathcal{T}, f^y_\upsilon(AI)))$. Thus, $f^y_\rho(f^y_\upsilon(AI))$ is restricted with respect to $\mathcal{T}$ and, by Lemma 15, it may not occur in the restricted chase of a program of the form $\mathcal{P}(\langle\mathcal{T}, \mathcal{A}\rangle)$. Furthermore, by Definition 14, if $f^y_\rho(f^y_\upsilon(AI))$ is restricted, then every term of the form $f^y_\rho(f^y_\upsilon(c))$, where c is a constant, is also restricted.

With Definition 14 and Lemma 15 in place, we proceed with the definition of a procedure to construct an overchase for some given TBox $\mathcal{T}$.

Definition 17. *Let $\mathcal{T}$ be a TBox. We define $\mathcal{V}_\mathcal{T}$ as the set initially containing every fact in $\mathcal{I}_\star(\mathcal{R}_\mathcal{T})$ which is then expanded by repeatedly applying the rules in Figure 3 (in non-deterministic order).*

Lemma 18. *The set $\mathcal{V}_\mathcal{T}$ is an overchase of the TBox $\mathcal{T}$.*

Proof. (Sketch) The lemma can be proven via induction on chase sequence of any ontology of the form $\mathcal{O} = \langle\mathcal{T}, \mathcal{A}\rangle$. Note that, $\mathcal{O}^0_R \subseteq \mathcal{V}_\mathcal{T}$ by the definition of $\mathcal{V}_\mathcal{T}$. It can be verified that, for every possible derivation of a set of facts during the computation of the chase of $\mathcal{O}$, such facts will always be contained in $\mathcal{V}_\mathcal{T}$.

Corollary 19. *The restricted chase of some TBox $\mathcal{T}$ terminates if $\mathcal{V}_\mathcal{T}$ is finite.*

Example 20. Let $\mathcal{T}$ be the TBox from Example 5. Then $\mathcal{V}_\mathcal{T}$ is as follows.

$$\mathcal{V}_\mathcal{T} = \{Film(\star), isProdBy(\star), Producer(\star), prod(\star, \star), \\ isProdBy(\star, f^y_\rho(\star)), Producer(f^y_\rho(\star)), prod(\star, f^y_\upsilon(\star)), Producer(f^y_\upsilon(\star))\}$$

Note that terms $f^y_\rho(f^y_\upsilon(\star))$ and $f^y_\upsilon(f^y_\rho(\star))$ are restricted and thus, they are not included in $\mathcal{V}_\mathcal{T}$. Since $\mathcal{V}_\mathcal{T}$ is finite, we can conclude termination of the restricted chase of the TBox $\mathcal{T}$.

In the previous example, we were able to ascertain termination of the restricted chase of $\mathcal{T}$ after verifying that the set $\mathcal{V}_\mathcal{T}$ is finite. A sufficient condition for finiteness of $\mathcal{V}_\mathcal{T}$ is to only allow cyclic terms up to a certain depth in this set. We use such condition to formally define RCA_n.

Definition 21. *A TBox $\mathcal{T}$ is* RCA_n *if and only if there are no n-cyclic terms in $\mathcal{V}_\mathcal{T}$. An ontology $\langle\mathcal{T},\mathcal{A}\rangle$ is RCA_n if and only if $\mathcal{T}$ is RCA_n.*

Theorem 22. *If a TBox $\mathcal{T}$ is RCA_n then the restricted chase of $\mathcal{T}$ terminates.*

We proceed with several results regarding the complexity of deciding RCA_n membership and reasoning over RCA_n ontologies.

Theorem 23. *Deciding whether some TBox $\mathcal{T}$ is RCA_n is in* ExpTime.

Theorem 24. *Let $\mathcal{O} = \langle\mathcal{T},\mathcal{A}\rangle$ be some RCA_n ontology and γ a query. Then, checking whether $\mathcal{O} \models \gamma$ is* ExpTime*-complete.*

To close the section, we present several results in which we theoretically compare the generality of RCA_n to $\text{MFA}^\cup$.

Theorem 25. *$MFA^\cup$ does not cover RCA_1.*

Proof. The TBox $\mathcal{T}$ from Example 5 is RCA_1 but not $\text{MFA}^\cup$.

Theorem 26. *If $\mathcal{T}$ is $MFA^\cup$ then $\mathcal{T}$ is RCA_n for every $n > |\mathcal{T}^\exists|$ where $\mathcal{T}^\exists$ is the set of all existential axioms in $\mathcal{T}$.*

6 Evaluation

6.1 An Empirical Comparison of RCA_n and $\text{MFA}^\cup$

In this section we include an empirical comparison of the generality of RCA_n and $\text{MFA}^\cup$. For our experiments, we use the TBoxes of the ontologies in the OWL Reasoner Evaluation workshop (ORE, `https://www.w3.org/community/owled/ore-2015-workshop/`) and Ontology Design Patterns (ODP, `http://www.ontologydesignpatterns.org`) datasets. The former is a large repository used in the ORE competition containing a large corpus of ontologies. The latter contains a wide range of smaller ontologies that capture design patterns commonly used in ontology modeling. The ORE dataset is rather large, and thus we restrict our experiments to the 294 ontologies with the smallest number of existential axioms, while skipping the 77 ontologies with the largest number of existential axioms. The number of such axioms contained in an ontology is a useful metric to predict the "hardness" of acyclicity membership tests; i.e. running these experiments would be very time-intensive, while our results, reported

ORE

∃-Axioms	Avg. Size	Count	$\text{MFA}^{\cup}$	RCA_1	RCA_2	RCA_3
1-5	175	70	70.0	87.1	92.9	92.9
6-10	219	48	58.3	83.3	83.3	83.3
11-25	916	54	83.3	85.2	91	91
26-100	521	42	54.8	59.5	61.9	61.9
101-500	1290	42	26.2	26.2	28.6	28.6
501-1922	5052	38	60.5	60.5	60.5	60.5
1-1922	1362	294	60.9	70.1	73.1	73.1

ODP

∃-Axioms	Size	Total	$\text{MFA}^{\cup}$	RCA_1	RCA_2	RCA_3
1-12	39	18	73.7	100.0	100.0	100.0

Fig. 4. Results for the ORE and ODP Repositories.

below, already indicate that for such very hard TBoxes $\text{MFA}^{\cup}$ and RCA_n will likely not differ much (while they differ significantly for ontologies with a lower count of existential axioms).

Only *Horn-$\mathcal{SRIQ}$* TBoxes which cannot be expressed in any of the OWL 2 profiles were considered in our experiments. This is because all OWL 2 RL TBoxes are acyclic (with respect to every applicable acyclicity notion known to us), and there already exist effective algorithms and efficient implementations that solve CQ answering over OWL 2 EL and OWL 2 QL ontologies [11, 17, 18] (albeit, if these do not include complex roles).

The results from our experiments are summarized in Figure 4. The evaluated TBoxes are sorted into brackets depending on the number of existential axioms they contain. For each bracket we provide the average number of axioms in the ontologies ("Avg. Size"), the number of ontologies ("Count"), and, for every condition "X" considered, the percentage of "X acyclic" ontologies

RCA_2 and RCA_3 turned out to be indistinguishable with respect to the TBoxes considered and thus, we limit our evaluation to RCA_n with $n \leq 3$. Our tests reveal that RCA_2 is significantly more general than $\text{MFA}^{\cup}$, particularly when it comes to TBoxes with a low count of existential axioms. However note that reasoning over ontologies with few (existential) axioms is in general not trivial: All of the ontologies considered in our materialization tests (see Figure 5) contain less than 20 existential axioms. For TBoxes containing from 1 to 10 existential axioms in the ORE dataset, more than half of the ontologies which are not $\text{MFA}^{\cup}$ are RCA_2. Furthermore, the 4 ontologies in the ODP dataset which are not $\text{MFA}^{\cup}$ are RCA_2. Interestingly, in both repositories we could not find any ontology that is $\text{MFA}^{\cup}$ but not RCA_1. Thus, with respect to the TBoxes in our corpus, RCA_1 already proves to be more general than $\text{MFA}^{\cup}$.

In total, we looked at 312 ontologies, 62% and 75% of which are $\text{MFA}^{\cup}$ and RCA_2, respectively. To gauge the significance of this improvement, we roughly compare these numbers with the results presented in [6]. In that paper, the authors consider a total of 336 ontologies, of which 49%, 58% and 68% are

Triples	Restricted		Oblivious		PAGOdA		Konc.
Count	C	Q1-Q4	C	Q1-Q4	P	Q1-Q4	R
2.8M	10	0 0 0 0	45	0 0 TO 0	89	OM 4 1 0	75
5.1M	21	0 0 0 0	138	0 0 TO 3	147	OM 1 2 0	214
6.7M	28	0 0 0 0	1029	2 0 TO 0	203	OM 2 3 1	506
8.1M	36	37 0 0 0	TO	- - - -	263	OM 2 2 6	1347
9.0M	37	0 0 0 0	OM	- - - -	113	1 1 1 1	198
17.8M	72	0 0 0 0	OM	- - - -	232	2 2 3 3	987
26.2M	107	0 0 0 0	OM	- - - -	378	4 10 12 5	3491
33.9M	141	0 1 0 0	OM	- - - -	521	6 21 21 12	TO
2.8M	8	0 0 0 1	70	0 0 0 74	51	OM 0 0 0	51
5.7M	16	0 0 0 2	158	1 1 1 154	99	OM 1 1 0	118
8.4M	26	0 0 0 3	242	1 1 2 186	142	OM 2 1 1	220
11.4M	37	1 0 0 5	341	2 2 3 311	197	OM 3 1 1	315
2.2M	11	0 0 0 0	56	0 0 0 1	61	28 0 TO 1	53
4.5M	27	2 0 0 0	133	0 0 1 2	121	60 0 TO 2	125
6.6M	42	3 1 1 0	216	1 1 2 3	186	TO 0 TO 5	292
8.9M	58	5 1 2 1	310	1 2 4 6	260	TO 0 TO 5	644

Fig. 5. Results for Reactome, Uniprot, LUBM and UOBM (sorted from top to bottom in the above table).

weakly acyclic [7], *jointly acyclic* [12] and MFA$^{\cup}$, respectively. Even though the comparison is not over the same TBoxes, we verify that the improvement in generality of our notion is in line with previous iterations of related work.

6.2 A Materialization Based Reasoner

We now report on an implementation of the restricted chase as defined in Section 3. Moreover, we also present an implementation of the oblivious chase with singularization, i.e., the chase as it must be used if we employ MFA$^{\cup}$ (see Section 4). We use the datalog engine RDFOx [15] in both implementations.

We evaluate the performance of our chase based implementations against Konclude [19], a very efficient OWL DL reasoner, and PAGOdA [20], a hybrid approach to query answering over ontologies. PAGOdA combines a datalog reasoner with a fully-fledged OWL 2 reasoner in order to provide scalable 'pay-as-you-go' performance and is, to the best of our knowledge, the only other implementation that may solve CQ answering over *Horn-$\mathcal{SRIQ}$* ontologies with completeness guarantees, albeit only in some cases. Nevertheless, PAGOdA was able to solve all the queries (that is, all of which for which it did not time-out or run out of memory) in this evaluation in a sound and complete manner.

We consider two real-world ontologies in our experiments, Reactome and Uniprot, and two standard benchmarks, LUBM and UOBM, all of which contain a large amount of ABox axioms. Axioms in these ontologies which are not expressible in *Horn-$\mathcal{SRIQ}$* were pruned. Furthermore, one extra axiom had to be removed from Uniprot for it to be both MFA$^{\cup}$ and RCA$_1$ acyclic.

The results from our experiments are summarized in Figure 5. For each ontology, we consider four samples of the original ABox. The number of triples contained in each one of these is indicated at the beginning of each row, under the column "Triples Count". As previously mentioned, we consider four different implementations: These include the two aforementioned variants of the chase ("Restricted" and "Oblivious"), PAGOdA ("PAGOdA") and Konclude ("Konc."). For both chase based implementations, we check the time it takes to compute the chase ("C") and then the time to solve each of the four queries crafted for each ontology ("Q1-Q4"). In a similar manner, we list the time PAGOdA takes to preprocess each ontology ("P") plus the time it takes to answer the queries ("Q1-Q4"). Finally, we list the time Konclude takes to solve realization; i.e., the task of computing every fact of the form $A(a)$ entailed by an ontology (note that Konclude cannot solve arbitrary CQ answering). Time-outs, indicated with "TO," were set at 1 hour for materialization and 5 minutes for queries. We make use of the acronym "OM" to indicate that an out-of-memory error occurred. Sometimes, a time-out or an out of memory error prevents us from answering the queries: Such a situation is indicated with "-." All experiments were performed on a MacBook Pro with 8GB of RAM and a 2.4 GHz Intel Core i5 processor.

For each ontology, we consider four different queries which are listed in the App. Section **??** included in the extended technical report. A summarized description of these queries, in which we ignore unary predicates, can be found in Figure 6. For every ontology, the query Q1 is of the form $\exists x, y, z R(x, y) \wedge R(z, y)$ where R is an existentially quantified role occurring in the TBox. It appears that PAGOdA has trouble with this kind of query, whereas the chase based implementations efficiently solve it in all but one case. This is probably due to the design of the hybrid reasoner which considers under and over approximations to provide complete answers to CQ: It appears that queries as the one previously considered find a large number of matches in the upper bound which slows down the performance of this reasoner. Queries Q2, and Q3 and Q4 are acyclic and cyclic, respectively (a query is acyclic if the shape of its body is acyclic). Even though it is well-known that answering acyclic CQs can be reduced to satisfiability [5], we included such a type of query in our evaluation in an attempt to verify whether solving acyclic queries is simpler than cyclic queries (this is indeed the case theoretically). Nevertheless, our experiments do not reveal any significant differences.

First, note that computing the restricted chase employing renaming techniques to deal with equality is way more efficient than computing the oblivious chase with singularization. We conjecture that this is because the efficient built-in capabilities of RDFOx to deal with equality and the fact that the rules that result from the application of singularization are rather cumbersome. Second, see that our proposed algorithm is also superior to PAGOdA when it comes to CQ answering. Third, the implementation of the restricted chase outperforms the DL reasoner Konclude by an order of magnitude when it comes to solve materialization of the larger samples considered (note that, by computing the chase of

$\mathsf{q}_1(w, y) :\ \mathsf{pE}(w, z), \mathsf{pE}(y, z)$
$\mathsf{q}_2(x, z) :\ \mathsf{mPE}(z, w), \mathsf{mPE}(z, w), \mathsf{p}(y, z), \mathsf{pC}(x, y)$
$\mathsf{q}_3(x, z) :\ \mathsf{fL}(x, w), \mathsf{fL}(x, y), \mathsf{sIB}(w, z), \mathsf{sIB}(y, z)$
$\mathsf{q}_4(x, z) :\ \mathsf{p}(w, z), \mathsf{p}(y, z), \mathsf{pC}(x, w), \mathsf{pC}(x, y)$

$\mathsf{q}_1(x, y) :\ \mathsf{cC}(x, z), \mathsf{cC}(y, z)$
$\mathsf{q}_2(x) :\ \mathsf{tF}(w, x), \mathsf{lO}(x, y), \mathsf{d}(x, z)$
$\mathsf{q}_3(x) :\ \mathsf{tF}(w, y), \mathsf{tF}(w, x), \mathsf{d}(y, z), \mathsf{d}(x, z)$
$\mathsf{q}_4(x) :\ \mathsf{ll}(x, w), \mathsf{cC}(w, z), \mathsf{ll}(x, y), \mathsf{cC}(y, z)$

$\mathsf{q}_1(x, z) :\ \mathsf{wF}(x, y), \mathsf{wF}(z, y), \mathsf{pA}(x, z)$
$\mathsf{q}_2(x) :\ \mathsf{a}(x, y), \mathsf{tO}(y, z), \mathsf{mO}(y, w)$
$\mathsf{q}_3(x, z) :\ \mathsf{tO}(y, z), \mathsf{a}(x, y), \mathsf{tC}(x, z)$
$\mathsf{q}_4(x) :\ \mathsf{pA}(x, z), \mathsf{pA}(x, y), \mathsf{a}(z, y), \mathsf{mO}(z, w), \mathsf{mO}(y, w)$

$\mathsf{q}_1(x, y) :\ \mathsf{tC}(x, z), \mathsf{tC}(y, z)$
$\mathsf{q}_2(x) :\ \mathsf{tAO}(x, y), \mathsf{pA}(z, x), \mathsf{tC}(w, y), \mathsf{wF}(x, v)\}$
$\mathsf{q}_3(x, y) :\ \mathsf{iFO}(x, y), \mathsf{I}(x, z)$
$\mathsf{q}_4(x, y) :\ \mathsf{hDDF}(x, z), \mathsf{hDDF}(y, z), \mathsf{hMDF}(x, w), \mathsf{hMDF}(y, w), \mathsf{wF}(x, v), \mathsf{wF}(y, v)$

Fig. 6. Summarized queries for Reactome (top left), Uniprot (top right), LUBM (bottom left) and UOBM (bottom right).

a program we already solve materialization). It is clear that our implementation also scales much better than the OWL DL reasoner.

7 Conclusions and Future Work

We introduce a novel acyclicity notion for *Horn-$\mathcal{SRIQ}$* TBoxes and prove it to be, theoretically and empirically, more general than previously existing conditions [6]. To the best our knowledge, this is the first acyclicity notion (for ontologies or rules) which considers termination of the restricted chase algorithm. Moreover, our contribution is also relevant in practice: Based on our ideas, we produce an implementation which vastly outperforms state-of-the-art reasoners.

As future work, we plan to lift our acyclicity condition to the case of general rules; i.e., not only those resulting from the translation of *Horn-$\mathcal{SRIQ}$* TBoxes. We also intend to work on further optimizing our implementation of the RCA_n membership check and our restricted chase based algorithm.

Acknowledgements. We wish to thank Bernardo Cuenca Grau for extensive discussions on the subject and valuable feedback. This work was supported by the National Science Foundation under awards 1017255 *III: Small: TROn – Tractable Reasoning with Ontologies* and 1440202 *EarthCube Building Blocks: Collaborative Proposal: GeoLink – Leveraging Semantics and Linked Data for Data Sharing and Discovery in the Geosciences*; the *ERC grant 647289* and the *European Research Council grant CODA 647289.*

References

1. 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)

2. Baget, J., Garreau, F., Mugnier, M., Rocher, S.: Extending acyclicity notions for existential rules. In: ECAI 2014. Frontiers in Artificial Intelligence and Applications, vol. 263, pp. 39–44. IOS Press (2014)
3. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase. In: Brewka, G., Lang, J. (eds.) Proc. 11th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'08). pp. 70–80. AAAI Press (2008)
4. Calvanese, D., Eiter, T., Ortiz, M.: Answering regular path queries in expressive dls via alternating tree-automata. Information and Computation 237, 12 – 55 (2014)
5. Carral, D., Hitzler, P.: Extending description logic rules. In: ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7295, pp. 345–359. Springer (2012)
6. Cuenca Grau, B., Horrocks, I., Krötzsch, M., Kupke, C., Magka, D., Motik, B., Wang, Z.: Acyclicity notions for existential rules and their application to query answering in ontologies. JAIR 47, 741–808 (2013)
7. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. Theor. Comput. Sci. 336(1), 89–124 (2005)
8. Glimm, B., Lutz, C., Horrocks, I., Sattler, U.: Conjunctive query answering for the description logic SHIQ. J. Artif. Intell. Res. (JAIR) 31, 157–204 (2008)
9. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer. W3C Recommendation (27 October 2009), available at `http://www.w3.org/TR/owl2-primer/`
10. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible SROIQ. In: Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, United Kingdom, June 2-5, 2006. pp. 57–67. AAAI Press (2006)
11. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in dl-lite. In: Lin, F., Sattler, U., Truszczynski, M. (eds.) KR 2010. AAAI Press (2010)
12. Krötzsch, M., Rudolph, S.: Extending decidable existential rules by joining acyclicity and guardedness. In: IJCAI. pp. 963–968 (2011)
13. Krötzsch, M., Rudolph, S., Hitzler, P.: Complexities of Horn description logics. ACM Trans. Comp. Log. 14(1), 2:1–2:36 (2013)
14. Marnette, B.: Generalized schema-mappings: from termination to tractability. In: PODS. pp. 13–22 (2009)
15. Nenov, Y., Piro, R., Motik, B., Horrocks, I., Wu, Z., Banerjee, J.: Rdfox: A highly-scalable RDF store. In: The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9367, pp. 3–20. Springer (2015)
16. Rudolph, S., Glimm, B.: Nominals, inverses, counting, and conjunctive queries or: Why infinity is your friend! CoRR abs/1401.3849 (2014)
17. Stefanoni, G., Motik, B., Horrocks, I.: Introducing nominals to the combined query answering approaches for $\mathcal{EL}$. In: AAAI (2013)
18. Stefanoni, G., Motik, B., Krötzsch, M., Rudolph, S.: The complexity of answering conjunctive and navigational queries over OWL 2 EL knowledge bases. J. Artif. Intell. Res. (JAIR) 51, 645–705 (2014)
19. Steigmiller, A., Liebig, T., Glimm, B.: Konclude: System description. J. Web Sem. 27, 78–85 (2014)
20. Zhou, Y., Grau, B.C., Nenov, Y., Kaminski, M., Horrocks, I.: Pagoda: Pay-as-you-go ontology query answering with a datalog reasoner. J. Artif. Intell. Res. (JAIR) 54, 309–367 (2015)