Undefined 1 (20) 1–5
IOS Press

Query Answering over Contextualized RDF/OWL Knowledge with Expressive Bridge Rules: Decidable Classes [1]

Editor(s): Name Surname, University, Country
Solicited review(s): Name Surname, University, Country
Open review(s): Name Surname, University, Country

Mathew Joseph [a,b,*],
[a] *DKM unit, FBK-IRST, Via Sommarive 18, 38050, Trento, Italy*
[b] *DISI, University Of Trento, Via Sommarive 5, 38123, Trento, Italy*
E-mail: {mathew}@fbk.eu

Abstract. The recent outburst of context-dependent knowledge on the Semantic Web (SW) has led to the realization of the importance of the quads in the SW community. Quads, which extend a standard RDF triple, by adding a new parameter of the 'context' of an RDF triple, thus informs a reasoner to distinguish between the knowledge in various contexts. Although this distinction separates the triples in an RDF graph into various contexts, and allows the reasoning to be decoupled across various contexts, bridge rules need to be provided for inter-operating the knowledge across these contexts. We call a set of quads together with the bridge rules, a quad-system. In this paper, we discuss the problem of query answering over quad-systems with expressive bridge rules using a contextualized OWL-Horst semantics. We present various decidable classes of quad-systems on which query answering can be done using forward reasoning. Besides undecidability of the most general case, both data and combined complexity of query entailment has been established for the various classes derived.

Keywords: Contextualized Query Answering, Contextualized RDF/OWL knowledge bases, Multi-Context Systems, Quads, Query answering, Description Logics, Semantic Web

1. Introduction

One of the major recent changes in the semantic web community is the transformation from a *triple* to a *quad* as its primary knowledge carrier. This change, primarily brought by the realization of the importance of contextual approach to knowledge representation [1], has resulted in more and more triple stores becoming *quad* stores. Some of the popular quad-stores are 4store[1], Openlink Virtuoso [2], and some of the current popular triple stores like Sesame[3], Allegrograph[4] internally keep track of the context by storing arrays of four names (c, s, p, o) (further denoted as $c\colon (s, p, o)$), where c is a URI that stands for the context of the triple (s, p, o). Some of the recent initiatives in this direction have also extended existing formats like N-Triples to N-Quads. The latest Billion triples challenge datasets (BTC 2011 and 2012) have been both released in the N-Quads format.

[1]The work was done as part of the research project, Contextualized Knowledge Repositores (CKR) for the Semantic Web, funded by FBK-IRST, Trento, Italy.
*Corresponding author. E-mail: mathew@fbk.eu.

[1]http://4store.org
[2]http://virtuoso.openlinksw.com/rdf-quad-store/
[3]http://www.openrdf.org/
[4]http://www.franz.com/agraph/allegrograph/

One of the main benefits of quads over triples are that they allow users to specify various attributes of meta-knowledge that further qualify knowledge [2], and also allow users to query for this meta knowledge [3]. Examples of these attributes, which are also called *context dimensions* [4], are provenance, creator, intended user, creation time, validity time, geo-location, and topic. Having defined various contexts in which triples are dispersed, one can declare in a meta-context mc, statements such as mc: (c_1, creator, John), mc: (c_1, expiryTime, "jun-2013") that talk about the knowledge in context c_1, in this case its creator and expiry time. Another benefit of such a contextualized approach is that it opens possibilities of interesting ways for querying a contextualized knowledge base. For instance, if context c_1 contains knowledge about football world cup 2010 and context c_2 about football euro cup 2012. Then the query "who beat spain in both euro cup 2012 and world cup 2010" can be formalized as the conjunctive query:

$$c_1\colon (x, \text{beat}, \text{Spain}) \wedge c_2\colon (x, \text{beat}, \text{Spain}),$$

where x is a variable. As the knowledge can be separated context wise and simultaneously be fed to separate reasoning engines, this approach not only increases efficiency and scalability, but also prevents reasoning inconsistencies in knowledge, especially in those circumstances, when the knowledge is derived from automated extraction/integration processes from heterogeneous sources. Examples of such inter-contextual inconsistencies are $\{c_1\colon (a, \texttt{owl:sameAs}, b), c_2\colon (a, \texttt{owl:differentFrom}, b)\}$, and $\{c_1\colon (C, \texttt{owl:disjointWith}, D), c_1\colon (a, \texttt{rdf:type}, C), c_2\colon (a, \texttt{rdf:type}, D)\}$. Note that in the above examples, knowledge in each context when considered separately is consistent. Although current reasoning engines like Sesame, 4store implicitly/explicitly support contexts, they do not do separate the triples in different contexts during the reasoning process, but runs reasoning procedures on the union of the triples in all the contexts. Whereas in our approach, knowledge in each context is treated separately during reasoning, and *bridge rules* like in DDL [5] are provided for enabling inter-operability of reasoning in different contexts. Such rules are primarily of the form:

$$c : \phi \rightarrow c' : \phi'$$

where ϕ, ϕ' are concepts/roles, c, c' are contexts. The bridge rules we consider, in this work, are an extension of the bridge rules in DDL [5] and *lifting rules* by McCarthy [7], in terms of expressivity, as we allow conjunctions and existential quantifiers in them.

In this work, we study contextual reasoning and query answering on contextualized RDF/OWL knowledge. We provide a basic semantics for contextual reasoning based on which we provide procedures for conjunctive query answering. For query answering, we use the notion of a *distributed chase*, which is an extension of a standard *chase* [17,18] that is widely used in databases and KR for the same. As far as semantics for reasoning is concerned, we adopt the approach given in works such as Distributed Description Logics [5], E-connections [19], and two-dimensional logic of contexts [20], which is to use a set of interpretation structures as a model for contextualized knowledge. In this way, knowledge in each context is separately interpreted to a different interpretation structure. The main contributions of this work are:

1. Adopting the approaches in the existing works mentioned above, we extend the standard OWL-Horst semantics to a context-based semantics that can be used for reasoning over contextualized RDF/OWL knowledge.
2. Studying conjunctive query answering over quad-systems, we show that it is undecidable for the most general class of quad-systems called *unrestricted quad-systems*.
3. We propose a decidable class of unrestricted quad-systems called *safe quad-systems*, for which we give both data and combined complexities of conjunctive query entailment. We also present an algorithm to decide whether an input quad-system is safe or not.
4. We further derive less expressive Horn-based fragments, for which we give both data and combined complexity results.

The paper is structured as follows. In section 2, we formalize the idea of contextualized quad-systems, giving various definitions and notations for setting the background. In section 3, we formalize the problem of query answering on quad-systems, define notions such as distributed chase that is further used for query answering, and give the undecidability results of query entailment on unrestricted quad-systems. In section 4, we present *safe* quad-systems and its properties. In section 5, the Horn based quad-systems. We provide a detailed discussion to other relevant related works in section 6, and conclude in section 7.

2. Contextualized Quad-Systems

Let $\mathbf{U}$ be the set of URIs, $\mathbf{B}$ the set of blank nodes, and $\mathbf{L}$ the set of literals. The set $\mathbf{C} = \mathbf{U} \cup \mathbf{B} \cup \mathbf{L}$ are called the set of (RDF) constants. Any $(s, p, o) \in \mathbf{C} \times \mathbf{C} \times \mathbf{C}$ is called a generalized RDF triple (from now on, just triple). A graph is defined as a set of triples. A *Quad* is a tuple of the form $c\colon (s, p, o)$, where (s, p, o) is a triple and c is a URI, called the *context identifier* that denotes the context of the RDF triple. Let X be the set of variables, any element of the set $\mathbf{C}^x = X \cup \mathbf{C}$ is a *term*. Any $(s, p, o) \in \mathbf{C}^x \times \mathbf{C}^x \times \mathbf{C}^x$ is called a *triple pattern*, and an expression of the form $c\colon (s, p, o)$, where (s, p, o) is a triple pattern, c a context identifier, is called a *quad pattern*. A *quad-graph* is defined as a set of quads. For any quad-graph Q and any context identifier c, we denote by $graph_Q(c)$ the set $\{(s, p, o) | c\colon (s, p, o) \in Q\}$. We denote by $Q_\mathcal{C}$ the quad-graph whose set of context identifiers is $\mathcal{C}$. For the sake of enabling interoperability between knowledge in different contexts, special rules called Bridge rules have to be provided:

Bridge rules (BRs) Formally, a BR is of the form:

$$\forall \vec{x} \forall \vec{z}\, [c_1\colon t_1(\vec{x}, \vec{z}) \wedge ... \wedge c_n\colon t_n(\vec{x}, \vec{z}) \rightarrow \exists \vec{y}\, c'_1\colon t'_1(\vec{x}, \vec{y}) \wedge ... \wedge c'_m\colon t'_m(\vec{x}, \vec{y})] \quad (1)$$

where $c_1, ..., c_n, c'_1, ..., c'_m$ are context identifiers, $\vec{x}, \vec{y}, \vec{z}$ are vectors of variables, $t_1(\vec{x}, \vec{z})$, ..., $t_n(\vec{x}, \vec{z})$ are triple patterns whose set of variables are from $\vec{x}$ and $\vec{z}$, $t'_1(\vec{x}, \vec{y}), ..., t'_m(\vec{x}, \vec{y})$ are triple patterns whose set of variables are from $\vec{x}$ and $\vec{y}$. For any BR, r, of the form (1), we use the notation $body(r) = \{c_1\colon t_1(\vec{x}, \vec{z}), ..., c_n\colon t_n(\vec{x}, \vec{z})\}$, and $head(r) = \{c'_1\colon t'_1(\vec{x}, \vec{y}), ..., c'_m\colon t'_m(\vec{x}, \vec{y})\}$.

Definition 2.1 (Quad-System)**.** *A* quad-system $QS_\mathcal{C}$ *is defined as a pair $\langle Q_\mathcal{C}, R\rangle$, where $Q_\mathcal{C}$ is a quad-graph, whose set of context identifiers is $\mathcal{C}$, and R is a set of BRs.*

For any quad-graph $Q_\mathcal{C}$, its size $|Q_\mathcal{C}|$ is the number of quads in $Q_\mathcal{C}$, and for any set of BRs r, its size $|r|$ is given by number of quad-patterns in r. For a set of BRs R, its size $|R|$ is given as $\Sigma_{r \in R}|r|$. For any quad-system $QS_\mathcal{C} = \langle Q_\mathcal{C}, R\rangle$, its size $|QS_\mathcal{C}| = |Q_\mathcal{C}| + |R|$.

Semantics We build our contextual semantics on top of OWL-Horst semantics. Readers should note that our system can be ported to the standard OWL semantics [41] with out much hassles, but the complexity results and finiteness properties of the quad-system fragments, which we define further, does not carry over, if one assumes the OWL semantics in the following definition of a quad-system model (definition 2.3). An OWL-Horst interpretation structure is a tuple $\langle$IR, IP, IC, IEXT, ICEXT, IS, LV$\rangle$, where IR is the object domain, IP $\subseteq$ IR is the property domain, IC $\subseteq$ IR is the class domain, IEXT, the property extension function, ICEXT, the class extension function, IS, the term interpretation function, and LV $\subseteq$ IR, are the set of literal values, with a list of additional semantic restrictions [24]. For details about OWL-Horst semantics and its computational properties, we refer the reader to appendix A. The semantics is defined using a distributed interpretation structure, which is an indexed set of OWL-Horst interpretation structures, defined as:

Definition 2.2 (Distributed Interpretation Structure)**.** *Given a quad graph $Q_\mathcal{C}$, a distributed interpretation structure is $\mathcal{I}^\mathcal{C} = \{I^c\}_{c \in \mathcal{C}}$ where $I^c = \langle IR^c, IP^c, IC^c, IEXT^c, ICEXT^c, IS^c, LV^c\rangle$ is an OWL-Horst interpretation structure.*

For any triple-pattern (s, p, o), and for any function σ, we use the notation $(s, p, o)[\sigma]$ to denote $(\sigma(s), \sigma(p), \sigma(o))$. We define the satisfaction relation, denoted by $\models$, between a distributed interpretation structure $\mathcal{I}^\mathcal{C}$ and a quad-system $QS_\mathcal{C}$ as:

Definition 2.3 (Model of a Quad-System)**.** *A distributed interpretation structure $\mathcal{I}^\mathcal{C} = \{I^c\}_{c \in \mathcal{C}}$ satisfies a quad-system $QS_\mathcal{C} = \langle Q_\mathcal{C}, R\rangle$, $\mathcal{I}^\mathcal{C} \models QS_\mathcal{C}$, iff all the following conditions are satisfied:*

1. *$I^c \models_{\textit{owl-horst}} graph_{Q_\mathcal{C}}(c)$ for each $c \in \mathcal{C}$, where $\models_{\textit{owl-horst}}$ is the classical satisfaction relation between an OWL-Horst interpretation and a graph.*
2. *$IS^{c_i}(a) = IS^{c_j}(a)$, if $IS^{c_i}(a) \in IR^{c_i}$ and $IS^{c_j}(a) \in IR^{c_j}$, for any $a \in \mathbf{C}$, $c_i, c_j \in \mathcal{C}$*
3. *for each BR $r \in R$ of the form (1) and for each $\sigma \in \Sigma$, if*

$$I^{c_1} \models_{\textit{owl-horst}} t_1(\vec{x}, \vec{z})[\sigma], ..., I^{c_n} \models_{\textit{owl-horst}} t_n(\vec{x}, \vec{z})[\sigma],$$

then there exists function $\sigma' \supseteq \sigma$, such that

$$I^{c'_1} \models_{\textit{owl-horst}} t'_1(\vec{x}, \vec{y})[\sigma'], ..., I^{c'_m} \models_{\textit{owl-horst}} t'_m(\vec{x}, \vec{y})[\sigma'],$$

where Σ be the set of all functions $\sigma : \mathbf{C}^x \rightarrow \mathbf{C}$, such that $\sigma(c) = c$, for any $c \in \mathbf{C}$.

Condition 1 in the above definition ensures that for any model $\mathcal{I}^\mathcal{C}$ of a quad-graph, each $I^c \in \mathcal{I}^\mathcal{C}$ is an OWL-Horst model of the set of triples in context c. Condition 2 ensures that, as for the standard RDF graphs, any constant c represents the same resource across a quad-graph, irrespective of the context in which it occurs. Condition 3 ensure that any model of a quad-system satisfies each BR in it. Any $\mathcal{I}^{\mathcal{C}}$ such that $\mathcal{I}^{\mathcal{C}} \models QS_{\mathcal{C}}$ is said to be a model of $QS_{\mathcal{C}}$. A quad-system $QS_{\mathcal{C}}$ is said to be *consistent* if there exists a model $\mathcal{I}^{\mathcal{C}}$, such that $\mathcal{I}^{\mathcal{C}} \models QS_{\mathcal{C}}$, and $QS_{\mathcal{C}}$ is said to be *inconsistent* if it is not consistent. For any quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$, it can be the case that $graph_{Q_{\mathcal{C}}}(c)$ is OWL-Horst consistent, for each $c \in \mathcal{C}$, whereas $QS_{\mathcal{C}}$ is not consistent. This is because the set of BRs R adds more knowledge to the quad-system, and restricts the set of models that satisfy the quad-system. One advantage of basing our semantics based on OWL-Horst semantics is that it is now possible to reason on quad extensions of both RDF and OWL (Full) ontologies.

Similar to the entailment of triples by a normal RDF graph, one can define the entailment of quads by a quad-graph as follows:

Definition 2.4 (Quad-system entailment). *A quad-system $QS_{\mathcal{C}}$ entails a quad $c\colon (s,p,o)$, in symbols $QS_{\mathcal{C}} \models c\colon (s,p,o)$, iff for any distributed interpretation structure $\mathcal{I}^{\mathcal{C}}$, if $\mathcal{I}^{\mathcal{C}} \models QS_{\mathcal{C}}$ then $\mathcal{I}^{\mathcal{C}} \models \langle\{c\colon (s,p,o)\}, \emptyset\rangle$. A quad-system $QS_{\mathcal{C}}$ entails a quad-graph $Q'_{\mathcal{C}'}$, in symbols $QS_{\mathcal{C}} \models Q'_{\mathcal{C}'}$ iff $QS_{\mathcal{C}} \models c\colon (s,p,o)$ for any $c\colon (s,p,o) \in Q'_{\mathcal{C}'}$. A quad-system $QS_{\mathcal{C}}$ entails a BR r iff for any distributed interpretation structure $\mathcal{I}^{\mathcal{C}}$, if $\mathcal{I}^{\mathcal{C}} \models QS_{\mathcal{C}}$ then $\mathcal{I}^{\mathcal{C}} \models \langle\emptyset, \{r\}\rangle$. For a set of BRs R, $QS_{\mathcal{C}} \models R$ iff $QS_{\mathcal{C}} \models r$, for every $r \in R$. Finally, a quad-system $QS_{\mathcal{C}}$ entails another quad-system $QS'_{\mathcal{C}'} = \langle Q'_{\mathcal{C}'}, R'\rangle$, in symbols $QS_{\mathcal{C}} \models QS'_{\mathcal{C}'}$ iff $QS_{\mathcal{C}} \models Q'_{\mathcal{C}'}$ and $QS_{\mathcal{C}} \models R'$.*

3. Query Answering on Quad-Systems

In this work, we limit ourselves to *Conjunctive Queries* (CQs), which are often called select-project-join queries. For any vector $\vec{x}$, let $|\vec{x}|$ denote its size. A CQ, $Q(\vec{x}) \leftarrow \exists \vec{y}\ t_1(\vec{x},\vec{y}) \wedge ... \wedge t_n(\vec{x},\vec{y})$, where t_i, for $i = 1,...,n$ are triple patterns over vectors of variables $\vec{x} = \langle x_1, ..., x_{|\vec{x}|}\rangle$ and $\vec{y} = \langle y_1, ..., y_{|\vec{y}|}\rangle$. The variables in $\vec{x}$ are called *free variables*, the variables in $\vec{y}$ are quantified variables. Let $\vec{a}$ be a vector such that $a_i \in \mathbf{U} \cup \mathbf{L}$ and $|\vec{x}| = |\vec{a}|$; then, $\vec{x}/\vec{a}$ denote simultaneous substitution of x_i by a_i, for $i = 1, ..., |\vec{x}|$. For any query $Q(\vec{x})$, $\vec{x}/\vec{a}$ in $Q(\vec{x})$ is denoted by $Q(\vec{a})$. Any non-boolean query $Q(\vec{x})$ becomes a boolean query after the substitution of $\vec{x}$ by a tuple of names, $\vec{a}$, of the same size.

For a quad-system, CQs are slightly extended to include context identifiers; we call such queries *Contextualized Conjunctive Queries* (CCQs). A CCQ $CQ(\vec{x})$ is an expression of the form:

$$CQ(\vec{x}) \leftarrow \exists \vec{y}\ c_1 : t_1(\vec{x},\vec{y}) \wedge ... \wedge c_n : t_n(\vec{x},\vec{y}) \quad (2)$$

where c_i are context identifiers, t_i are triple patterns over vectors of variables $\vec{x}$ and $\vec{y}$, for $i = 1, ..., n$. Intuitively, $c_i : t_i(\vec{x},\vec{y})$ is a query that has to be propagated to context c_i, for $i = 1, ..., n$. As for the CQs, for any CCQ $CQ(\vec{x})$, $CQ(\vec{a})$ is boolean.

For any distributed interpretation structure $\mathcal{I}^{\mathcal{C}} = \{I^{c_i}\}_{c_i \in \mathcal{C}}$ with $I^{c_i} = \langle \mathrm{IR}^{c_i}, \mathrm{IP}^{c_i}, \mathrm{IC}^{c_i}, \mathrm{IEXT}^{c_i}, \mathrm{ICEXT}^{c_i}, \mathrm{LV}^{c_i}, \mathrm{IS}^{c_i}\rangle$, let $\mathrm{IR}^{\mathcal{C}} = \bigcup_{c_i \in \mathcal{C}} \mathrm{IR}^{c_i}$ be called the domain of $\mathcal{I}^{\mathcal{C}}$. A vector $\vec{a}$ is an *answer* for a CCQ $CQ(\vec{x})$ w.r.t. a distributed interpretation structure $\mathcal{I}^{\mathcal{C}} = \{I^{c_i}\}_{c_i \in \mathcal{C}}$, in symbols $\mathcal{I}^{\mathcal{C}} \models CQ(\vec{a})$, iff $I^{c_i} \models_{\text{owl-horst}} t_i(\vec{a},\vec{y})[\mu]$, for $i = 1, ..., n$, where $\mu : \{y_1, ..., y_{|\vec{y}|}\} \rightarrow \mathrm{IR}^{\mathcal{C}}$ is an assignment from set of variables in $\vec{y}$ to the domain of $\mathcal{I}^{\mathcal{C}}$. A vector $\vec{a}$ is a *certain answer* for a CCQ $CQ(\vec{x})$ w.r.t. a quad-system $QS_{\mathcal{C}}$ iff $\mathcal{I}^{\mathcal{C}} \models CQ(\vec{a})$ for every model $\mathcal{I}^{\mathcal{C}}$ of $QS_{\mathcal{C}}$. In this case, we say that $QS_{\mathcal{C}}$ *entails* $CQ(\vec{a})$. Note that the problem of deciding, for any given $CQ(\vec{x})$, vector $\vec{a}$, and a quad-system $QS_{\mathcal{C}}$, if $QS_{\mathcal{C}} \models CQ(\vec{a})$ is called the *CCQ entailment problem*, and is the problem primarily studied in this paper. Since $CQ(\vec{a})$ is boolean, w.l.o.g., assume that input CCQ is boolean, and focus on the boolean CCQ entailment problem. In order to do query answering over a quad-system, we employ what has been called in the literature, a *chase* [17,18], specifically, we adopt the notion of the skolem chase in Marnette [25]. For any OWL ontology O, and for any boolean CQ $Q(\vec{a})$, its chase $chase(O)$ has the property: $O \models Q(\vec{a})$ iff $t(\vec{a},\vec{y})[\mu] \in chase(O)$, for all $t(\vec{a},\vec{y}) \in Q(\vec{a})$, where $\mu : \vec{y} \rightarrow \mathbf{C}$. We extend the notion of chase to a quad-system, which we call a distributed chase, abbreviated as *dChase*. In the following, we show how the dChase of a quad-system can be constructed.

3.1. dChase of a Quad-System

For any BR r, we apply *skolemization* that replaces $\vec{y}$ in r with $\vec{f}^r$, where $\vec{f}^r = \langle f_1^r, ..., f_{|\vec{y}|}^r\rangle$, is vector of globally unique Skolem functions such that each $f_i^r : \mathbf{C}^{|\vec{x}|} \rightarrow \mathbf{B}_i^r$, $\mathbf{B}_i^r \subseteq \mathbf{B}$ is a fresh set of blank nodes. Intuitively, $f_i^r(\vec{x})$ gives a fresh blank node for every distinct input vector $\vec{a}$. For any BR r defined before, we omit universal quantifiers, and replace conjunctions with commas (Datalog notation), and r is written as:

$$c_1 : t_1(\vec{x}, \vec{z}), ..., c_n : t_n(\vec{x}, \vec{z}) \rightarrow$$
$$c'_1 : t'_1(\vec{x}, \vec{f}^r(\vec{x})), ..., c'_m : t'_m(\vec{x}, \vec{f}^r(\vec{x})) \quad (3)$$

Normalization: after skolemization, a BR can be treated like a horn first-order formula and can be transformed to a semantically equivalent set of formulas, such that there is only a single quad-pattern in the head part of the BR. For example, result of this transformation on a skolemized BR of the form (3), is the following:

$$c_1 : t_1(\vec{x}, \vec{z}), ..., c_n : t_n(\vec{x}, \vec{z}) \rightarrow c'_1 : t'_1(\vec{x}, \vec{f}^r(\vec{x}))$$
$$...$$
$$c_1 : t_1(\vec{x}, \vec{z}), ..., c_n : t_n(\vec{x}, \vec{z}) \rightarrow c'_m : t'_m(\vec{x}, \vec{f}^r(\vec{x}))$$

It can be noted that this transformation is linear, and hence w.l.o.g. we assume that for any set of BRs R, its skolemization $sk(R)$ is also normalized in the above fashion.

Let M be the set of all functions, such that each $\mu \in M$ is a function from the set of variables X to the set of constants C. For any quad-graph $Q_\mathcal{C}$ and BR r of the form (3), *application* of r on $Q_\mathcal{C}$, denoted by $r(Q_\mathcal{C})$, is given as:

$$r(Q_\mathcal{C}) = \bigcup_{\mu \in M} \left\{ \begin{array}{l} c'_1 : t'_1(\mu(\vec{x}), \vec{f}^r(\mu(\vec{x})), ..., c'_m : t'_m(\\ \mu(\vec{x}), \vec{f}^r(\mu(\vec{x}))|\ c_1 : t_1(\mu(\vec{x}), \mu(\vec{z})), \\ ..., c_n : t_n(\mu(\vec{x}), \mu(\vec{z})) \in Q_\mathcal{C} \end{array} \right\}$$

For any set of rules R, application of R on $Q_\mathcal{C}$ is given as:

$$R(Q_\mathcal{C}) = \bigcup_{r \in R} r(Q_\mathcal{C}),$$

for any quad-graph $Q_\mathcal{C}$, we define:

$$\mathit{owl\text{-}horst\text{-}closure}(Q_\mathcal{C}) = \textstyle\bigcup_{c \in \mathcal{C}} \{c\colon (s,p,o)$$
$$|\ (s,p,o) \in \mathit{owl\text{-}horst\text{-}closure}(graph_{Q_\mathcal{C}}(c))\}$$

For any quad-system $QS_\mathcal{C} = \langle Q_\mathcal{C}, R \rangle$, let R_F be the skolemization of the set of rules in R with existential quantifiers, called as *generating BRs*, and $R_I = R - R_F$, called as *non-generating BRs*. Let $dChase_0(QS_\mathcal{C}) = \mathit{owl\text{-}horst\text{-}closure}(Q_\mathcal{C})$,

$dChase_{i+1}(QS_\mathcal{C}) = \mathit{owl\text{-}horst\text{-}closure}(dChase_i(QS_\mathcal{C}) \cup R_I(dChase_i(QS_\mathcal{C})))$, if $R_I(dChase_i(QS_\mathcal{C})) \not\subseteq dChase_i(QS_\mathcal{C})$;

$dChase_{i+1}(QS_\mathcal{C}) = \mathit{owl\text{-}horst\text{-}closure}(dChase_i(QS_\mathcal{C}) \cup R_F(dChase_i(QS_\mathcal{C})))$, otherwise.

dChase of $QS_\mathcal{C}$, denoted by $dChase(QS_\mathcal{C})$, is given as:

$$dChase(QS_\mathcal{C}) = \bigcup_{i \in \mathbb{N}} dChase_i(QS_\mathcal{C})$$

It can be noted that, if there exists i such that $dChase_i(QS_\mathcal{C}) = dChase_{i+1}(QS_\mathcal{C})$, then, $dChase(QS_\mathcal{C}) = dChase_i(QS_\mathcal{C})$. Any iteration i, such that $dChase_i(QS_\mathcal{C})$ is computed by the application of the set of (non-)generating BRs, R_F (resp. R_I), on $dChase_{i-1}(QS_\mathcal{C})$ is called a *generating iteration* (resp. *non-generating iteration*).

In general, for any quad-system $QS_\mathcal{C} = \langle Q_\mathcal{C}, R \rangle$, its dChase need not be unique, since final constructed dChase depends on the order in which rules in R are applied and the order in which the assignments to a BR are applied. By ordering the set of constants and variables (for instance, lexicographically), one can also use this to order the set of quads in $Q_\mathcal{C}$ and rules in R. In each dChase iteration, applying BRs respecting this order, and also for each BR r, applying assignments to r in this order, one can construct a unique dChase for any quad-system. From now on, we assume that for any quad-system $QS_\mathcal{C}$, $dChase(QS_\mathcal{C})$ denotes its unique dChase constructed using the above mentioned procedure. We call the sequence $dChase_0(QS_\mathcal{C}), dChase_1(QS_\mathcal{C}), ...$, the *dChase sequence* of $QS_\mathcal{C}$. The following lemma shows that, for any quad-system the result of a single generating iteration and any subsequent non-generating iterations in its dChase sequence, causes only a worst case exponential blow up in size.

Lemma 3.1. *For a quad-system $QS_\mathcal{C} = \langle Q_\mathcal{C}, R \rangle$, then the following holds: (i) if $i \in \mathbb{N}$ is a generating iteration, then $|dChase_i(QS_\mathcal{C})| = \mathcal{O}(|dChase_{i-1}(QS_\mathcal{C})|^{|R|})$, (ii) suppose $i \in \mathbb{N}$ is a generating iteration, and for any $j \geq 1$, $i+1$, ..., $i+j$ are non-generating iterations, then $|dChase_{i+j}(QS_\mathcal{C})| = \mathcal{O}(|dChase_{i-1}(QS_\mathcal{C})|^{|R|})$, (iii) for any iteration k, $dChase_k(QS_\mathcal{C})$ can be computed in time $\mathcal{O}(|dChase_{k-1}(QS_\mathcal{C})|^{|R|})$.*

Proof. (sketch)

(i) R can be applied on $dChase_{i-1}(QS_\mathcal{C})$ by grounding R to the set of constants in $dChase_{i-1}(QS_\mathcal{C})$, the number of such groundings is of the order $\mathcal{O}(|dChase_{i-1}(QS_\mathcal{C})|^{|R|})$, $|R(dChase_{i-1}(QS_\mathcal{C}))| = \mathcal{O}(|R| * |dChase_{i-1}(QS_\mathcal{C})|^{|R|})$. Since OWL-horst closure only increases the size polynomially [24], $|dChase_i(QS_\mathcal{C})| = \mathcal{O}(|dChase_{i-1}(QS_\mathcal{C})|^{|R|})$.

(ii) From (i) we know that $|R(dChase_{i-1}(QS_\mathcal{C}))| = \mathcal{O}(|dChase_{i-1}(QS_\mathcal{C})|^{|R|})$. Since, no new constant is introduced in any subsequent non-generating iterations, and since any quad contains only four constants, the set of constants in any subsequent dChase iteration is given by $\mathcal{O}(4 * |dChase_{i-1}(QS_\mathcal{C})|^{|R|})$. Since only these many constants can appear in positions c, s, p, o of any quad generated in the subsequent iterations, the size of $dChase_{i+j}(QS_\mathcal{C})$ can only increase polynomially, which means that $|dChase_{i+j}(QS_\mathcal{C})| = \mathcal{O}(|dChase_{i-1}(QS_\mathcal{C})|^{|R|})$.

(iii) Since any dChase iteration k involves the following two operations: (a) $owl\text{-}horst\text{-}closure()$, (b) computing $R(dChase_{k-1}(QS_\mathcal{C}))$. (a) can be done in PTIME w.r.t to its input [24]. (b) can be done in the following manner: ground R to the set of constants in $dChase_{i-1}(QS_\mathcal{C})$; then for each grounding g, if $body(g) \subseteq dChase_{i-1}(QS_\mathcal{C})$, then add $head(g)$ to $R(dChase_{k-1}(QS_\mathcal{C}))$. Since, the number of such groundings is of the order $\mathcal{O}(|dChase_{k-1}(QS_\mathcal{C})|^{|R|})$, and checking, if each grounding is contained in $dChase_{k-1}(QS_\mathcal{C})$, can be done in time polynomial in $dChase_{k-1}(QS_\mathcal{C})$, the time taken for (b) is $\mathcal{O}(|dChase_{k-1}(QS_\mathcal{C})|^{|R|})$. Hence, any iteration k can be done in time $\mathcal{O}(|dChase_{k-1}(QS_\mathcal{C})|^{|R|})$. □

In the following, we give a few computational characteristics of quad-systems whose BRs are of the form (1), which we call *unrestricted quad-systems*. It turns out the dChase computation for unrestricted quad-systems is some times impossible, as the dChase can be infinite. This raises the question if there are other approaches that can be used, for instance similar problem arises in DLs with value creation, due to the presence of existential quantifiers, whereas the approaches like the one in Glim et al. [26] provides an algorithm for CQ entailment based on query rewriting. On a close look, it can be observed that a quad-system can be seen as a set of Datalog+/- rules [6] using ternary predicates, one for each context whose instances are the set of triples in the contexts. Although, query entailment is known to become undecidable on adding Datalog like rules to DLs with value creation, see for instance SWRL [27], and is also undecidable for general Datalog+/- rules, we are not aware of any works that provide undecidability results for our bounded 3-arity case, or the undecidability of adding rules to DLs with out value creation like OWL 2 RL, RDF, or in our case OWL-Horst. The following theorem establishes the fact the CCQ entailment problem for unrestricted quad-systems is undecidable.

Theorem 3.2. *The CCQ entailment problem over unrestricted quad-systems is undecidable.*

Proof. (sketch) We show that the well known undecidable problem of non-emptiness of intersection of context-free grammars (CFGs) is reducible to the CCQ entailment problem. Given two CFGs, $G_1 = \langle V_1, T, S_1, P_1\rangle$ and $G_2 = \langle V_2, T, S_2, P_2\rangle$, where V_1, V_2 are the set of variables, T such that $T \cap (V_1 \cup V_2) = \emptyset$ is the set of terminals. $S_1 \in V_1$ is the start symbol of G_1, and P_1 are the set of PRs of the form $v \to \vec{w}$, where $v \in V$, $\vec{w}$ is a sequence of the form $w_1...w_n$, where $w_i \in V_1 \cup T$. Similarly s_2, P_2 is defined. Deciding whether the language generated by the grammars $L(G_1)$ and $L(G_2)$ have non-empty intersection is known to be undecidable [32].

Given two CFGs, $G_1 = \langle V_1, T, S_1, P_1\rangle$ and $G_2 = \langle V_2, T, S_2, P_2\rangle$, we encode grammars G_1, G_2 into a quad-system of the form $QS_c = \langle Q_c, R\rangle$, with a single context identifier, c. Each PR $r = v \to \vec{w} \in P_1 \cup P_2$, with $\vec{w} = w_1w_2w_3..w_n$, is encoded as a BR of the form:

$$c\colon (x_1, w_1, x_2), c\colon (x_2, w_2, x_3), ..., c\colon (x_n, w_n, x_{n+1}) \\ \to c\colon (x_1, v, x_{n+1})$$

where $x_1, .., x_{n+1}$ are variables. For each terminal symbol $t_i \in T$, R contains a BR of the form:

$$c\colon (x, \texttt{rdf:type}, C) \to \exists y\, c\colon (x, t_i, y), \\ c\colon (y, \texttt{rdf:type}, C)$$

and Q_c contains only the triple:

$$c\colon (a, \texttt{rdf:type}, C)$$

It can be observed that:

$$QS_c \models \exists y\, c\colon (a, S_1, y) \;\wedge\; c\colon (a, S_2, y) \leftrightarrow \\ L(G_1) \cap L(G_2) \neq \emptyset$$

We refer the reader to Appendix for the complete proof. □

4. Safe Quad-Systems: A decidable class

In this section, we define a more general fragment of quad-systems, in which we put some restrictions on the blank nodes generated in the dChase, in order to guarantee decidability and finiteness of dChase. We start by giving some necessary notations.

The set of constants occurring in a quad-graph $Q_{\mathcal{C}}$, given as $\mathbf{C}(Q_{\mathcal{C}}) = \{c, s, p, o \mid c\colon (s, p, o) \in Q_{\mathcal{C}}\}$. The set of URIs in $Q_{\mathcal{C}}$, is given by $\mathbf{U}(Q_{\mathcal{C}}) = \mathbf{C}(Q_{\mathcal{C}}) \cap \mathbf{U}$. The set of blank nodes $\mathbf{B}(Q_{\mathcal{C}})$, the set of literals $\mathbf{L}(Q_{\mathcal{C}})$ are similarly defined. For a BR r, the set of terms in r, is given as:

$$\mathbf{C}^x(r) = \{c, s, p, o \mid c\colon (s, p, o) \in body(r) \cup head(r)\}$$

The set of terms in a set of BRs R is given by $\mathbf{C}^x(R) = \bigcup_{r \in R} \mathbf{C}^x(r)$. The URIs, blank nodes, literals, and variables in a set of BRs R are similarly defined, and are denoted as $\mathbf{U}(R), \mathbf{B}(R), \mathbf{L}(R), X(R)$, respectively. For any quad-system, $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$, the set of constants in $QS_{\mathcal{C}}$ is given by $\mathbf{C}(QS_{\mathcal{C}}) = \mathbf{C}(Q_{\mathcal{C}}) \cup \mathbf{C}(R)$. The sets $\mathbf{U}(QS_{\mathcal{C}}), \mathbf{B}(QS_{\mathcal{C}}), \mathbf{L}(QS_{\mathcal{C}})$, and $X(QS_{\mathcal{C}})$ are similarly defined for any quad-system $QS_{\mathcal{C}}$.

For any quad-system $QS_{\mathcal{C}}$, the set of blank-nodes $\mathbf{B}(dChase(QS_{\mathcal{C}}))$ in its $dChase(QS_{\mathcal{C}})$, not only contains blank nodes in $\mathbf{B}(QS_{\mathcal{C}})$, but can also contain blank nodes, that are generated by Skolem functions during the dChase construction process. We call such blank nodes, *Skolem blank nodes* of $dChase(QS_{\mathcal{C}})$ and is given as $\mathbf{B}_{sk}(dChase(QS_{\mathcal{C}})) = \mathbf{B}(dChase(QS_{\mathcal{C}})) \setminus \mathbf{B}(QS_{\mathcal{C}})$. A quad in $dChase(QS_{\mathcal{C}})$ that contains a Skolem blank node is called a *Skolem quad*. Any Skolem blank node b can uniquely be represented by the expression, $f(\vec{k})$, where f is the Skolem function symbol and $\vec{k}$ the vector of constants used by f to generate b. Extending this also to the set of constants in $dChase(QS_{\mathcal{C}})$, and recursively expanding each $k \in \vec{k}$, one can define, for each constant k in $dChase(QS_{\mathcal{C}})$, its generating expression:

Definition 4.1 (genExp)**.** *For any constant $k \in dChase(QS_{\mathcal{C}})$, its generating expression $genExp(k)$ is defined inductively as:*

- $genExp(k) = k$, *for any* $k \in \mathbf{C}(QS_{\mathcal{C}})$,
- $genExp(k) = f(genExp(k_1), ..., genExp(k_n))$, *if* $k \in \mathbf{B}_{sk}(dChase(QS_{\mathcal{C}}))$, *generated by a Skolem function f using the vector of constants* $\langle k_1, ..., k_n\rangle$.

For any Skolem blank node $b = f(k_1, ..., k_n)$, where $k_1, ..., k_n$ are constants, we denote this relation between k_i to b with the relational symbol *childOf*. Moreover, since children of a Skolem blank node can be Skolem blank nodes, which themselves can have children, one can naturally define relation *descendantOf*=childOf$^+$ as the transitive closure of childOf.

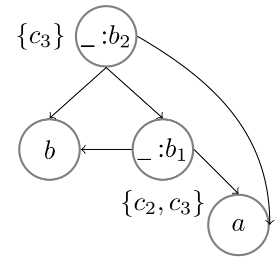

Fig. 1. descendance graph of $\_:b_2$

Example 4.2. Consider the quad-system $\langle Q_{\mathcal{C}}, R\rangle$, where $Q_{\mathcal{C}} = \{c_1\colon (a, b, c)\}$, and suppose the skolemization $sk(R)$ of R is the following set:

$$sk(R) = \left\{ \begin{array}{r} c_1{:}(x_1, x_2, x_3) \to c_2{:}(x_1, x_2, f_1(x_1, x_2)), \\ c_3{:}(x_1, x_2, f_1(x_1, x_2)) \\ c_2{:}(x_4, x_5, x_6) \to c_3{:}(f_2(x_5, x_6), x_5, x_6) \end{array} \right\}$$

Iterations during dChase construction are:

$$\begin{aligned} dChase_0(QS_{\mathcal{C}}) &= \{c_1{:}(a, b, c)\} \\ dChase_1(QS_{\mathcal{C}}) &= \{c_1{:}(a, b, c), c_2(a, b, \_:b_1), \\ &\qquad c_3(a, b, \_:b_1)\} \\ dChase_2(QS_{\mathcal{C}}) &= \{c_1{:}(a, b, c), c_2(a, b, \_:b_1), \\ &\qquad c_3(a, b, \_:b_1), c_3{:}(\_:b_2, b, \_:b_1)\}, \\ dChase(QS_{\mathcal{C}}) &= dChase_2(QS_{\mathcal{C}}), \end{aligned}$$

where $\_:b_1 = f_1(a, b)$ and $\_:b_2 = f_2(b, \_:b_1)$, p.s. we have not shown the quads derived using local OWL-Horst inferencing. Note that $genExp(\_:b_1) = f_1(a, b)$ and $genExp(\_:b_2) = f_2(b, f_1(a, b))$. For any Skolem blank node its descendant hierarchy can be visualized using a *descendance graph*. Descendance graph for $\_:b_2$ is shown in Fig.1. Before defining the notion of safety, let us define the notion of origin-contexts for any skolem blank-node.

Definition 4.3 (Origin-contexts)**.** *For any quad-system $QS_{\mathcal{C}}$, and for any Skolem blank node $b \in \mathbf{B}_{sk}(dChase(QS_{\mathcal{C}}))$, its origin-contexts, is given as $originContexts(b) = \{c \mid \exists i.c{:}(s, p, o) \in dChase_i(QS_{\mathcal{C}}),\ s = b \vee p = b \vee o = b$, and $\nexists j < i, \exists c'.c'{:}(s', p', o') \in dChase_j(QS_{\mathcal{C}}), s' = b \vee p' = b \vee o' = b\}$.*

Intuitively, origin-contexts for a Skolem blank node b is the set of contexts in which triples containing b are first generated, during dChase construction. Note that there can be multiple contexts to which b can simultaneously be generated. For the quad-system and dChase, presented in example 4.2,

$originContexts(\_:b_1) = \{c_2, c_3\}, originContexts(\_:b_2) = \{c_3\}$. Note that in Fig. 1, the origin-contexts of $\_:b_1$ and $\_:b_2$ are shown along with their node labels.

Definition 4.4 ((Un)safe quad-systems)**.** *A quad-system $QS_\mathcal{C}$ is said to be unsafe, if its dChase $dChase(QS_\mathcal{C})$, contains blank nodes $b \neq b'$, with $b, b' \in \mathbf{B}_{sk}(dChase(QS_\mathcal{C}))$, such that b is a descendant of b' and $originContexts(b) = originContexts(b')$. A quad-system is safe iff it is not unsafe.*

Intuitively, a quad-system is safe, if there does not exist a Skolem blank-node that is generated in a (set of) context(s), using another Skolem blank-node generated in the same (set of) context(s). Safe quad-systems in this way prevents recursive generation of blank nodes generated in a (set of) context(s) using blank nodes that are generated in the same (set of) context(s). One should note that unsafety is an approximation of infinite dChases, for which dChase computation is non-terminating. It was shown in Deutsch et al. [28] that the decision problem of deciding whether, for any set of rules with existential variables, its chase is finite or not is undecidable, in general. As we have seen earlier, for any quad-system $QS_\mathcal{C} = \langle Q_\mathcal{C}, R\rangle$, whose dChase is $dChase(QS_\mathcal{C})$, any $b \in \mathbf{B}_{sk}(dChase(QS_\mathcal{C}))$ can be visualized using its descendance graph, that is rooted at b. Furthermore, the descendance graph has the following property:

Property 4.5 (DAG property)**.** *For a safe quad-system $QS_\mathcal{C}$, and for any blank node $b \in \mathbf{B}_{sk}(dChase(QS_\mathcal{C}))$, its descendance graph is a directed acyclic graph (DAG).*

Proof. By construction, as there exists no descendant for any constant $k \in \mathbf{C}(QS_\mathcal{C})$, there cannot be any out-going edge from any such k. Hence, any member of $\mathbf{C}(QS_\mathcal{C})$ cannot be involved in cycles. Hence, the only members that can be involved can be the members of $\mathbf{C}(dChase(QS_\mathcal{C})) - \mathbf{C}(QS_\mathcal{C}) = \mathbf{B}_{sk}(dChase(QS_\mathcal{C}))$. But if there exists a $b \in \mathbf{B}_{sk}(dChase(QS_\mathcal{C}))$, such that there exists a cycle through b, then this implies that b is a descendant of b. Since this would violate the safety property, and imply that $QS_\mathcal{C}$ is unsafe, which is a contradiction. □

Since the descendance graph G of any Skolem blank node $b \in \mathbf{B}_{sk}(dChase(QS_\mathcal{C}))$ is rooted at b and there are no cycles in G, any path from b terminates at some node. Hence, one can use a tree traversal technique, such as preOrder (visit a node first and then its children), to sequentially traverse each node in G. The algorithm 1 below, takes a descendance graph G and unravels it into a tree. The algorithm first removes all the transitive edges from G, i.e. if there are $v, v', v'' \in V$, with $(v, v'), (v', v''), (v, v'') \in E$, then it removes (v, v''). Note that the information that v'' is a descedant of v is still present in the new graph. The algorithm then traverses the graph in preorder fashion, as it encounters a node v, if v has an indegree k greater than one, it splits v to k fresh nodes $v_1, ..., v_k$, and distributes the set of edges incident to v across $v_1, ..., v_k$, such that (i) each v_i has at-most one incoming edge (ii) all the edges incident to v are incident to some v_i. Whereas out going edges of v are retained for each v_i. Hence, after the splitting operation each v_i has an indegree 1, where as outdegree v_i is same as the out de-

Algorithm 1:

`UnRavel` (Descendance Graph $G = \langle V, E\rangle$, Label l)
`/* procedure to unravel, a descendance graph into a tree */`
Input : descendance graph G, partial function $l : V \rightarrow 2^\mathcal{C}$
Output: boolean value True or False
begin
  $G' = \langle V, E\rangle$=RemoveTranstiveEdges(G);
  foreach *Node* $v \in$ *preOrder*(G') **do**
    if $(k =$ *indegree*$(v)) > 1$ **then**
      $V' = V - \{v\} \cup \{v_1, ..., v_k\}$;`/* where each` v_i `is fresh */`
      for $i = 1; i \leq k; i$++ **do**
        $l(v_i) = l(v)$;`// copy the node and origin context labels`
      foreach $(v, v') \in E$ **do**
        $E' = E - \{(v, v')\} \cup \{(v_1, v'), ..., (v_k, v')\}$;
        `/* copy outgoing edge of` v `to each` v_i `*/`
      $i = 1$;
      foreach $(v', v) \in E$ **do**
        `/* distribute the incoming edges of` v`, one for each` v_i `*/`
        $E' = E - \{(v', v)\} \cup \{(v', v_i)\}$;
        i++;
      $E = E', V = V'$;

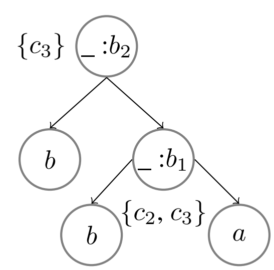

Fig. 2. descendance graph of Fig. 1 unraveled into a tree

gree of v. Hence, once all the nodes are visited, any node, except the root, in the new graph G has an in-degree 1. G is still rooted, connected, acyclic, and is hence a tree. The algorithm terminates as there are no cycles in graph, and at some point reaches a node with no children. For instance the unraveling of the descendance graph of _ :b_2 in Fig. 1 of example 4.2, is shown in Fig. 2. The following property holds for any Skolem blank node of a safe quad-system.

Property 4.6. *For a safe quad-system $QS_{\mathcal{C}}$, and any Skolem blank node in $dChase(QS_{\mathcal{C}})$, the unravelling (Algorithm 1) of its descendance graph results in a tree t such that:*

1. *any leaf node of t is from the set $\mathbf{C}(QS_{\mathcal{C}})$,*
2. *any non-leaf node of t is from the set $\mathbf{B}_{sk}(dChase(QS_{\mathcal{C}}))$,*
3. *$order(t) \leq max\{ar(f_i) | f_i$ is a Skolem function symbol occurring in $sk(R)\}$,*
4. *there cannot be a path between $b \neq b'$ with $originContexts(b) = originContexts(b')$.*

Proof.

1. Since any node n in the dependency graph is such that $n \in \mathbf{C}(dChase(QS_{\mathcal{C}}))$, and since $\mathbf{C}(dChase(QS_{\mathcal{C}})) = \mathbf{C}(QS_{\mathcal{C}}) \cup \mathbf{B}_{sk}(dChase(QS_{\mathcal{C}}))$. Since any member $m \in \mathbf{B}_{sk}(dChase(QS_{\mathcal{C}}))$ is generated from a skolem function and a set of constants, hence has at-least one child. Since n is a leaf node $n \in \mathbf{C}(QS_{\mathcal{C}})$.
2. Since any member $m \in \mathbf{C}(QS_{\mathcal{C}})$ cannot have descedants and since any non-leaf node has children, m cannot be a non-leaf node. Hence, non-leaf nodes should be from $\mathbf{B}_{sk}(dChase(QS_{\mathcal{C}}))$.
3. Since order of t is the out degree of a node n of t, such that there exists no other node n' such that $outdegree(n') > outdegree(n)$. Let n be any such node, but since n is a blank node, this implies that n is generated from a skolem function f occurring in $sk(R)$, which implies that $outdegree(n) = ar(f)$.
4. Since any path from b to b' implies that b' is a descendant of b, then it should be the case that $originContexts(b) \neq originContexts(b')$, otherwise safety condition would be violated.

□

The property above is exploited to show that there exists a finite bound in the dChase size and its computation time.

Lemma 4.7. *For any safe quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$, the following holds: (i) the dChase size $|dChase(QS_{\mathcal{C}})| = \mathcal{O}(2^{2^{2^{|QS_{\mathcal{C}}|}}})$, (ii) $dChase(QS_{\mathcal{C}})$ can be computed in* 3EXPTIME*, (iii) if $|R|$ and the set of schema triples in $Q_{\mathcal{C}}$ is fixed to a constant, then $|dChase(QS_{\mathcal{C}})|$ is a polynomial in $|QS_{\mathcal{C}}|$ and can be computed in* PTIME.

Proof. (sketch)

(i) Each Skolem blank node generated has a constrained tree structure t such that its depth is exponential in $\mathcal{C}$, since there cannot be paths in t that contain nodes with same $C \subseteq \mathcal{C}$ as origin-context labels. Also order of the tree is bounded by m, where m is the maximal arity of Skolem functions in $sk(R)$. Hence, any such tree can have $\mathcal{O}(m^{2^{|\mathcal{C}|}})$ leaf nodes and $\mathcal{O}(m^{2^{|\mathcal{C}|}})$ inner nodes, and since each of the children can be elements in $\mathbf{C}(QS_{\mathcal{C}})$, the number of such trees are clearly triple exponential in $\mathbf{C}(QS_{\mathcal{C}})$, hence bounds the number of Skolem blank nodes generated in dChase construction.

(ii) From (i) $|dChase(QS_{\mathcal{C}})|$ is triply exponential in $|QS_{\mathcal{C}}|$, and since each iteration add at-least one quad to its dChase, the number of iterations are bounded triple exponentially in $|QS_{\mathcal{C}}|$. Also, by lemma 3.1 any iteration i can be done in time $\mathcal{O}(|dChase_{i-1}(QS_{\mathcal{C}})|^{|R|})$. Since using (i) $|dChase_{i-1}(QS_{\mathcal{C}})| = \mathcal{O}(2^{2^{2^{|QS_{\mathcal{C}}|}}})$, each iteration i can be done in time $\mathcal{O}(2^{|R| * 2^{2^{|QS_{\mathcal{C}}|}}})$. Also, as number of iterations is triple exponential, computing $dChase(QS_{\mathcal{C}})$ is in 3EXPTIME.

(iii) Since $|R|$ is fixed to a constant, the set of skolem function symbols F in $sk(R)$, the arity of any $f \in F$, and set of origin contexts are constants. Because of this, the number of tree structures of skolem blank-nodes generated is a constant z. Hence, the number of inner nodes and leaves of any such tree, which can be taken by any constant in $\mathbf{C}(QS_{\mathcal{C}})$. Hence, the number of skolem blank nodes generated is $\mathcal{O}(|\mathbf{C}(QS_{\mathcal{C}})|^z)$. Hence, the set of constants in $dChase(QS_{\mathcal{C}})$ is a polynomial in $|QS_{\mathcal{C}}|$, and also is $|dChase(QS_{\mathcal{C}})|$.

Since in any dChase iteration except the final one, atleast one quad should be added, and also since the final dChase can have atmost $\mathcal{O}(|QS_{\mathcal{C}}|^z)$ triples, the total number of iterations are bounded by $\mathcal{O}(|QS_{\mathcal{C}}|^z)$ (†). By lemma 3.1, since any iteration i can be computed in $\mathcal{O}(|dChase_{i-1}(QS_{\mathcal{C}})|^{|R|}$ time, and since $|R|$ is a constant, the time required for each iteration is a polynomial in $|dChase_{i-1}(QS_{\mathcal{C}})|$, which is atmost a polynomial in $|QS_{\mathcal{C}}|$. Hence, any dChase iteration can be performed in polynomial time in size of $QS_{\mathcal{C}}$ (‡).

From (†) and (‡), it can be concluded that dChase can be computed in PTIME. □

Lemma 4.8. *For any safe quad-system, the following holds: (i) data complexity of CCQ entailment is in* PTIME, *(ii) combined complexity of CCQ entailment is in* 3EXPTIME.

Proof. Given a safe quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R \rangle$, since $dChase(QS_{\mathcal{C}})$ is finite, a boolean CCQ $CQ()$ can naively be evaluated by binding the set of constants in the dChase to the variables in the $CQ()$, and then checking if any of these bindings are contained in $dChase(QS_{\mathcal{C}})$. The number of such bindings can atmost be $|dChase(QS_{\mathcal{C}})|^{|CQ()|}$ (†).

(i) Since for data complexity, the size of the BRs $|R|$, the set of schema triples, and $|CQ()|$ is fixed to constant. From lemma 4.7 (iii), we know that under the above mentioned settings the dChase can be computed in PTIME and is polynomial in the size of $QS_{\mathcal{C}}$. Since $|CQ()|$ is fixed to a constant, and from (†), binding the set of constants in $dChase(QS_{\mathcal{C}})$ on $CQ()$ still gives a number of bindings that is worst case polynomial in the size of $|QS_{\mathcal{C}}|$. Since membership of these bindings can checked in the polynomially sized dChase in PTIME, the time required for CCQ entailment is in PTIME.

(ii) Since in this case $|dChase(QS_{\mathcal{C}})| = \mathcal{O}(2^{2^{2^{|QS_{\mathcal{C}}|}}})$ (‡), from (†) and (‡), binding the set of constants in $dChase(QS_{\mathcal{C}})$ to $CQ()$ amounts to $\mathcal{O}(2^{|CQ()|*2^{2^{|QS_{\mathcal{C}}|}}})$ bindings. Since the dChase is triple exponential in $|QS_{\mathcal{C}}|$, checking the membership of each of these bindings can be done in 3EXPTIME. Hence, the combined complexity is in 3EXPTIME. □

Theorem 4.9. *For any safe quad-system, the following holds: (i) The data complexity of CCQ entailment is* PTIME*-complete (ii) The combined complexity of CCQ entailment is in* 3EXPTIME*-complete.*

Proof. (i)(Membership) See lemma 4.8 for the membership in PTIME.
(Hardness) Follows from the PTIME-hardness of data complexity of CCQ entailment for Horn quad-systems (Theorem 5.2), which are contained in safe quad-systems.
(ii) (Membership) See lemma 4.8.
(Hardness) See following heading. □

4.1. 3EXPTIME-Hardness of CCQ Entailment

In this subsection, we show that the decision problem of CCQ entailment for safe quad-systems is 3EXPTIME-hard. We show this by reduction of the word-problem of a double-exponential space bounded alternating turing machine (ATM) [33] to the CCQ query entailment problem. From the following well known relation that gives the relation between the space complexity of ATMs to time complexity of deterministic turing machines (DTM):

$$\text{ASPACE}(f(n)) = \text{DTIME}(2^{\mathcal{O}(f(n))})$$

it follows that A2EXPSPACE=3EXPTIME. Hence, by reducing a problem word problem that is A2EXPSPACE-hard, it follows that CCQ entailment problem is 3EXPTIME-hard.

An ATM M is a tuple $M = \langle Q, \Sigma, \Delta, q_0 \rangle$, where

- $Q = U \uplus E$ is a disjoint union of a set of universal states U and existential states E,
- Σ is a finite alphabet that includes the blank symbol $\square$,
- $\Delta \subseteq (Q \times \Sigma) \times (Q \times \Sigma \times \{+1, -1\})$ is a transition relation
- $q_0 \in Q$ is the initial state.

A (universal/existential) configuration is a word $\vec{\alpha} \in \Sigma^* Q \Sigma^* (\Sigma^* U \Sigma^* / \Sigma^* E \Sigma^*)$. A configuration $\vec{\alpha}_2$ is a successor of the configuration $\vec{\alpha}_1$, if one of the following holds:

1. $\vec{\alpha}_1 = \vec{w}_l q \sigma \sigma_r \vec{w}_r$ and $\vec{\alpha}_2 = \vec{w}_l \sigma' q' \sigma_r \vec{w}_r$, if $(q, \sigma, q', \sigma', R) \in \Delta$, or
2. $\vec{\alpha}_1 = \vec{w}_l q \sigma$ and $\vec{\alpha}_2 = \vec{w}_l \sigma' q' \square$, if $(q, \sigma, q', \sigma', R) \in \Delta$, or
3. $\vec{\alpha}_1 = \vec{w}_l \sigma_l q \sigma \vec{w}_r$ and $\vec{\alpha}_2 = \vec{w}_l q' \sigma_l \sigma' \vec{w}_r$, if $(q, \sigma, q', \sigma', L) \in \Delta$.

where $q, q' \in Q$, $\sigma, \sigma', \sigma_l, \sigma_r \in \Sigma$, and $\vec{w}_l, \vec{w}_r \in \Sigma^*$. Suppose we put bound on the on the number of tape cells of an ATM by a value n, since each configuration $\vec{c}$ can be represented in size $|\vec{c}| = n + 1$, the number of possible configurations is bounded by $O(2^{n+1})$. A configuration $\vec{c} = \vec{w}_l q \vec{w}_r$ is an accepting configuration iff

- $q \in U$, and all successor configurations of $\vec{c}$ are accepting, or
- $q \in E$, and there exists a successor configuration of w that is accepting

Note that, by this definition, all the universal configurations with out any successors are trivially accepting, and existential configurations with out any successors are trivially non-accepting. A language $L \subseteq \Sigma^*$ is accepted by a double exponential space bounded ATM M, if for every $\vec{w} \in L$, M accepts w in space $\mathcal{O}(2^{2^{|\vec{w}|}})$.

Simulating ATMs using Safe Quad-Systems Consider an ATM $M = \langle Q = U \uplus E, \Sigma, \delta, q_0 \rangle$, and a string w, with $|w| = n$. Since the number of storage cells is doubly exponentially bounded, we first construct a quad-system $QS^M_{\mathcal{C}} = \langle Q^M_{\mathcal{C}}, R \rangle$, where $\mathcal{C} = \{c_0, c_1, ..., c_n\}$, note that $|\mathcal{C}| = |\vec{w}| + 1$. We employ a technique, that is adapted from Cali et al. [34], to iteratively generate a doubly exponential number of objects that represent the cells of the tape of the ATM. Let $Q^M_{\mathcal{C}}$ be initialized with the following quads:

$$c_0\colon (k_0, \texttt{rdf:type}, R), c_0\colon (k_1, \texttt{rdf:type}, R), c_0\colon (k_0, \texttt{rdf:type}, min_0), c_0\colon (k_1, \texttt{rdf:type}, max_0), c_0\colon (k_0, succ_0, k_1)$$

Now for each pair of elements of type R in c_i, a skolem blank-node is generated in c_{i+1}, and hence follows the recurrence relation $r(a) = a^2$, which after n iterations yields $r^n(a) = a^{2^n}$. In this way, a doubly exponential long chain of elements is created in c_n using the following set of rules:

$$c_i\colon (x_0, \texttt{rdf:type}, R), c_i\colon (x_1, \texttt{rdf:type}, R) \rightarrow \exists y\; c_{i+1}\colon (x_0, x_1, y), c_{i+1}\colon (y, \texttt{rdf:type}, R)$$

The combination of minimal element with the minimal element (elements of type min_i) in c_i create the minimal element in c_{i+1}, and similarly the combination of maximal element with the maximal element (elements of type max_i) in c_i create the maximal element of c_{i+1}

$$c_{i+1}\colon (x_0, x_0, x_1), c_i\colon (x_0, \texttt{rdf:type}, min_i) \rightarrow c_{i+1}\colon (x_1, \texttt{rdf:type}, min_{i+1})$$

$$c_{i+1}\colon (x_0, x_0, x_1), c_i\colon (x_0, \texttt{rdf:type}, max_i) \rightarrow c_{i+1}\colon (x_1, \texttt{rdf:type}, max_{i+1})$$

Successor relation $succ_{i+1}$ is created in c_{i+1} using the following set of rules, using the well-known, integer counting technique:

$$c_i\colon (x_1, succ_i, x_2), c_{i+1}\colon (x_0, x_1, x_3), c_{i+1}\colon (x_0, x_2, x_4) \rightarrow c_{i+1}\colon (x_3, succ_{i+1}, x_4)$$

$$c_i\colon (x_1, succ_i, x_2), c_{i+1}\colon (x_1, x_3, x_5), c_{i+1}\colon (x_2, x_4, x_6), c_i\colon (x_3, \texttt{rdf:type}, max_i), c_i\colon (x_4, \texttt{rdf:type}, min_i) \rightarrow c_{i+1}\colon (x_5, succ_{i+1}, x_6)$$

By virtue of the first rule below, each of the objects representing the cells of the ATM are linearly ordered by the relation $succ$. Also the transitive closure of $succ$ is defined using relation $succt$

$$c_n\colon (x_0, succ_n, x_1) \rightarrow c_n\colon (x_0, succ, x_1)$$

$$c_n\colon (x_0, succ, x_1) \rightarrow c_n\colon (x_0, succt, x_1)$$

$$c_n\colon (x_0, succt, x_1), c_n\colon (x_1, succt, x_2) \rightarrow c_n\colon (x_0, succt, x_2)$$

Each of the above set rules are instantiated for $0 \le i < n$, and hence in this way after n generating dChase iterations, c_n has doubly exponential number of elements of type R, that are ordered linearly using the relation $succ$. Various triple patterns that are used to encode the possible configurations, runs and their relations in M are:

$(x_0, head, x_1)$ denotes the fact that in configuration x_0, the head of the ATM is at cell x_1.
$(x_0, state, x_1)$ denotes the fact that in configuration x_0, the ATM is in state x_1.
(x_0, σ, x_1), for each $\sigma \in \Sigma$, which denote the fact that in configuration x_0, the cell x_1 contains σ.
$(x_0, succ, x_1)$ denotes the linear order between cells of the tape.
$(x_0, succt, x_1)$ denotes the transitive closure of $succ$.
$(x_0, conSucc_\delta, x_1)$ to denote the fact that x_1 is a successor configuration of x_0 by the transition δ.
$(x_0, \mathrm{rdf:type}, Accept)$ denotes the fact that the configuration x_0 is an accepting configuration.

Since in our construction, each $\sigma \in \Sigma$ is represented as relation, one should constrain that no two alpabets $\sigma \neq \sigma'$ are on the same cell, we encode this using the following set of axioms

$$c_n\colon (\sigma, \texttt{owl:disjointWith}, \sigma'),\ \text{for } \sigma \neq \sigma' \in \Sigma$$

Note that `owl:disjointWith` relation is present even in weak logics such as OWL-Horst.

Initialization Suppose the initial configuration $I_0 = q_0\vec{w}\Box$, where $\vec{w} = \sigma_0...\sigma_{n-1}$, then we enode this using the constant I_0 and the following rules in our quad-system $QS^M_{\mathcal{C}}$ as:

$$c_n\colon (x, \texttt{rdf:type}, min_n) \rightarrow c_n\colon (I_0, head, x),$$
$$c_n\colon (I_0, state, q_0)$$

$$c_n\colon (x_0, \texttt{rdf:type}, min_n) \wedge \bigwedge_{i=0}^{n-1} c_n\colon (x_i, succ, x_{i+1})$$
$$\rightarrow \bigwedge_{i=0}^{n-1} c_n\colon (I_0, \sigma_i, x_i) \wedge c_n\colon (I_0, \Box, x_n)$$

$$c_n\colon (I_0, \Box, x_0), c_n\colon (x_0, succt, x_1) \rightarrow c_n\colon (I_0, \Box, x_1)$$

The last rule copies the $\Box$ to every succeeding cell.

Transitions For every left transition $\delta = (q_j, \sigma', -1) \in \Delta(q, \sigma)$, the following rules:

$$c_n\colon (x_0, head, x_i), c_n\colon (x_0, \sigma, x_i), c_n\colon (x_0, state, q),$$
$$c_n\colon (x_j, succ, x_i) \rightarrow \exists y\ c_n\colon (x_0, conSucc_\delta, y),$$
$$c_n\colon (y, head, x_j), c_n\colon (y, \sigma', x_i), c_n\colon (y, state, q_j)$$

For every right transition $\delta = (q_j, \sigma', +1) \in \Delta(q, \sigma)$, the following rules:

$$c_n\colon (x_0, head, x_i), c_n\colon (x_0, \sigma, x_i), c_n\colon (x_0, state, q),$$
$$c_n\colon (x_i, succ, x_j) \rightarrow \exists y\ c_n\colon (x_0, conSucc_\delta, y),$$
$$c_n\colon (y, head, x_j), c_n\colon (y, \sigma', x_i), c_n\colon (y, state, q_j)$$

Inertia If in any configuration the head is at position i of the tape, then in every successor configuration, elements in preceeding and following positions i of the tape are retained. The following two rules ensures this

$$c_n\colon (x_0, head, x_i), c_n\colon (x_0, conSucc_\delta, x_1),$$
$$c_n\colon (x_j, succt, x_i), c_n\colon (x_0, \sigma, x_j) \rightarrow c_n\colon (x_1, \sigma, x_j)$$

$$c_n\colon (x_0, head, x_i), c_n\colon (x_0, conSucc_\delta, x_1),$$
$$c_n\colon (x_i, succt, x_j), c_n\colon (x_0, \sigma, x_j) \rightarrow c_n\colon (x_1, \sigma, x_j)$$

The rules above are instantiated for every $\sigma \in \Sigma$ and for every $\delta \in \Delta(q, \sigma)$, for $q \in Q, \sigma \in \Sigma$

Acceptance For each existential states $q_e \in E$,

$$c_n\colon (x_0, state, q_e), c_n\colon (x_0, conSucc_\delta, x_1), c_n\colon (x_1,$$
$$\texttt{rdf:type}, Accept) \rightarrow c_n\colon (x_0, \texttt{rdf:type}, Accept)$$

For each universal state $q_u \in U$,

$$c_n\colon (x_0, state, q_u) \wedge \bigwedge_{\delta \in \Delta(q_u, \sigma)} (c_n\colon (x_0, conSucc_\delta, x_1),$$
$$c_n\colon (x_1, \texttt{rdf:type}, Accept)) \rightarrow$$
$$c_n\colon (x_0, \texttt{rdf:type}, Accept)$$

Finally since M accepts $\vec{w}$ iff only if the inital configuration $I_0 = q_0\vec{w}\Box$ is an accepting configuration. Hence, I_0 is accepting iff $QS^M_{\mathcal{C}} \models c\colon (I_0, \texttt{rdf:type}, Accept)$. Hence, CCQ entailment is 3EXPTIME-hard.

4.2. *Procedure for detecting safe quad-systems*

In this subsection, we present a procedure for deciding whether a given quad-system is safe or not. If the quad-system is safe, the result of procedure is a *safe dChase*, that contains the standard dChase, and can be used for query answering. Since safety property of a quad-system is attributed to the dChase of the quad-system, the procedure nevertheless performs the standard operations for computing the dChase, but also generate quads that indicate origin-contexts of each Skolem blank nodes generated. In each iteration, a test for safety is performed, by checking the presence of a Skolem blank-nodes that violates the safety condition. In case violation of safety condition is detected, a distinguished constant is generated and the dChase construction is aborted prematurely. On the contrary, if there exists an iteration i such that $dChase_i(QS_{\mathcal{C}}) = dChase_{i+1}(QS_{\mathcal{C}})$, the dChase computation stops with a completed dChase. Since all the additional quads produced for accounting information, uses a distinguished context identifier $c_c \notin \mathcal{C}$, the computed safe dChase itself can be used for standard query answering. We introduce a few notations and definitions.

Definition 4.10 (Context Scope)**.** *The context scope of a term t in a set of quad-patterns Q, denoted by $cScope(t, Q)$ is given as: $cScope(t, Q) = \{c \mid c\colon (s, p, o) \in Q, s = t \vee p = t \vee o = t\}$. For any vector $\vec{a}$ of terms, the context scope of $\vec{a}$ over Q is given as: $cScope(\vec{a}, Q) = \bigcup_{a \in \vec{a}} cScope(a, Q)$.*

For any BR r, of the form (3), in order to make the variables in r explicit, we also use the notation $body(r)(\vec{x}, \vec{z})$ and $head(r)(\vec{x}, \vec{y})$ for $body(r)$ and

$head(r)$, respectively. For any quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$, let $sk(R)$ be the skolemization of R and c_c be an arbitrary context identifier such that $c_c \notin \mathcal{C}$, then for $r \in sk(R)$, we define transformation $aug(r)$ as:

$$aug(r) = \bigwedge_{q \in body(r)(\vec{x},\vec{z})} q \rightarrow \bigwedge_{q' \in head(r)(\vec{x},\vec{f}(\vec{x}))} q' \ \wedge$$

$$\forall f_i(\vec{x}) \in \vec{f}(\vec{x}) \ [\bigwedge_{x_i \in \vec{x}} c_c\colon (x_i, \text{descendantOf}, f_i(\vec{x}))$$

$$\wedge\, c_c\colon (f_i(\vec{x}), \text{descendantOf}, f_i(\vec{x})) \ \wedge c_c\colon (f_i(\vec{x}),$$

$$\text{originContext}, c_i)]$$

Intuitively, the transformation aug, on a skolemized BR r whose set of Skolem function symbols is $\vec{f}(\vec{x})$, augments the head part of r with the following three additional types of quad patterns:

1. $c_c\colon (x_i, \text{descendantOf}, f_i(\vec{x}))$, for every Skolem function $f_i(\vec{x})$ in $\vec{f}(\vec{x})$ and universally quantified variable $x_i \in \vec{x}$. This is done because, during dChase computation, an application of a BR containing $\vec{f_i}(\vec{x})$, in which a vector $\vec{a}$ is assigned to $\vec{x}$, resulting in the generation of a Skolem blank node $f_i(\vec{a})$, any $a_i \in \vec{a}$ is a descendant of $f_i(\vec{a})$. Hence, due to these additional quad-patterns, quads of the form $c_c\colon (a_i, \text{descendantOf}, f_i(\vec{a}))$ are also produced, and in this way, keeps track of the descendants of any Skolem blank node produced.
2. $c_c\colon (f_i(\vec{x}), \text{ descendantOf}, \ f_i(\vec{x}))$, in order to maintain also the reflexivity of 'descendantOf' relation.
3. $c_c\colon (f_i(\vec{x}), \text{originContext}, c_i)$, for every Skolem function $f_i(\vec{x})$ in $\vec{f}(\vec{x})$, and for any c_i that is in the context scope of $f_i(\vec{x})$ in $head(r)(\vec{x}, \vec{f}(\vec{x})))$. This is done because, during dChase computation, an application of a BR contain $\vec{f_i}(\vec{x})$, in which a vector $\vec{a}$ is assigned to $\vec{x}$, resulting in the generation of a Skolem blank node $f_i(\vec{a})$, is produced in the set of contexts c_i in $cScope(f_i(\vec{x}), head(r)(\vec{x}, \vec{f}(\vec{x})))$. Hence, due to these additional quad-patterns, quads of the form $c_c\colon (f_i(\vec{a}), \text{originContext}, c_i)$ are also produced. In this way, keeps track of the origin-contexts of any Skolem blank node produced.

It can be noticed that for any BR r without any Skolem function symbols, the transformation aug leaves r unchanged. For any set of skolemized BRs R, let $aug(R) = \bigcup_{r \in R} aug(r)$. The function unSafeTest defined below, given a set of augmented BRs R and a quad-graph Q checks, if application of any $r \in R$ on Q violates the safety condition. **unSafeTest**(Q, R)=True iff $\exists r = body(r)(\vec{x},\ \vec{z}) \rightarrow head(r)\ (\vec{x}, \vec{f}(\vec{x})) \in R$, $\exists \mu \in M$, $\exists b, b' \in \mathbf{B}$, $\exists f_i(\vec{x}) \in \vec{f}(\vec{x})$ with the following being satisfied:

- $body(r)(\vec{x}, \vec{z})[\mu] \subseteq Q$, and
- $b \in \mu(\vec{x})$, and
- $c_c\colon (b', \text{descendantOf}, b) \in Q$, and
- $\{c \mid c_c\colon (b', \text{originContext}, c) \in Q\} = cScope(f_i(\vec{x}), head(r)(\vec{x}, \vec{f}(\vec{x})))$.

Intuitively, uncSafeTest returns True, if there is a BR $r \in R$, containing Skolem function symbols, with body $body(r)(\vec{x}, \vec{z})$, head $head(r)(\vec{x}, \vec{f}(\vec{x}))$, exists an assignment μ with Skolem blank node $b \in \mu(\vec{x})$, such that r is applicable on Q using μ, and when μ applied to r will produce a Skolem blank node b'', such that origin-contexts of b'' is equal to origin-contexts of b', which is a descendant of b. For a set of BRs R, the *safe application* of R on a quad-graph $Q_{\mathcal{C}}$ is defined as:

$$R^{\text{safe}}(Q_{\mathcal{C}}) = \begin{cases} \textbf{unSafe,} & \text{If unSafeTest}(R, Q_{\mathcal{C}}) = \text{True} \\ \bigcup_{r \in aug(R)} r(Q_{\mathcal{C}}), & \text{Otherwise} \end{cases}$$

where **unSafe** is a constant that is generated, if in any iteration, the safety condition is violated. For any quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$, we define its *safe dChase* $dChase^{\text{safe}}(QS_{\mathcal{C}})$ as follows:

$$dChase_0^{\text{safe}}(QS_{\mathcal{C}}) = owl\text{-}horst\text{-}closure(Q_{\mathcal{C}} \ \cup$$

$$c_c\colon (\text{descendentOf}, \texttt{rdf:type}, \texttt{owl:Transit-}$$

$$\texttt{-iveProperty}))$$

$$dChase_{i+1}^{\text{safe}}(QS_{\mathcal{C}}) = owl\text{-}horst\text{-}closure($$

$$dChase_i^{\text{safe}}(QS_{\mathcal{C}}) \cup \ R^{\text{safe}}(dChase_i^{\text{safe}}(QS_{\mathcal{C}})))$$

$$dChase^{\text{safe}}(QS_{\mathcal{C}}) = \bigcup_{i \in \mathbb{N}} dChase_i^{\text{safe}}(QS_{\mathcal{C}})$$

If there exists i such that
$dChase_i^{\text{safe}}(QS_{\mathcal{C}}) = dChase_{i+1}^{\text{safe}}(QS_{\mathcal{C}})$,
then,
$dChase^{\text{safe}}(QS_{\mathcal{C}}) = dChase_i^{\text{safe}}(QS_{\mathcal{C}})$.
The following theorem shows that the procedure above described for detecting unsafe quad-systems is sound and complete:

Theorem 4.11. *For any quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$, the constant* ***unSafe*** $\in dChase^{\textit{safe}}(QS_{\mathcal{C}})$, *iff $QS_{\mathcal{C}}$ is unsafe.*

Hence, after running above mentioned procedure, if constant **unSafe** is not generated, then its safe dChase itself can be used for CCQ answering, since dChase is contained in safe dChase, and all the quads generated for accounting information is in the distinguished context c_c. Hence, for any boolean CCQ that does not contain quads of the form $c_c\colon (s,p,o)$, dChase entails CCQ iff safe dChase entails CCQ.

5. Horn Quad-Systems: Restricting to Horn BRs

In this section, we investigate the complexity of CCQ answering on quad-systems, whose BRs do no have existential quantifiers, removing existential quantifiers from BRs of the form (1) results in:

$$c_1 : t_1(\vec{x},\vec{z}) \wedge ... \wedge c_n : t_n(\vec{x},\vec{z}) \rightarrow$$
$$c'_1 : t'_1(\vec{x}) \wedge ... \wedge c'_m : t'_m(\vec{x})$$

Since a logical formula of the form, $A \rightarrow B_1 \wedge ... \wedge B_n$ can be rewritten to a semantically equivalent set of formulas of the form, $A \rightarrow B_1, ..., A \rightarrow B_n$, the set of BRs R can be rewritten to R', such that each $r \in R'$ is the form:

$$c_1 : t_1(\vec{x},\vec{z}), ..., c_n : t_n(\vec{x},\vec{z}) \rightarrow\ c'_1 : t'_1(\vec{x}) \qquad (4)$$

Also R' is linear in size of R, and hence, w.l.o.g, we assume that each $r \in R$ is of the form (4). We call these rules *Horn bridge rules*, as they resemble the Horn rules in logic programming. We call a quad-system whose BRs are all of Horn-type, a *Horn quad-system*. Since there exists no existential variables in BRs of a Horn quad-system, the skolemization does not change R, i.e. $sk(R) = R$. Since there are no blank-node generating Skolem functions in $sk(R)$, no skolem blank nodes are produced during chase computation. Hence, there can be no violation of the safety condition in section 4, and hence, the class of horn quad-systems are contained in the class of safe quad-systems. Of course, this containment is strict as any quad-system that contains a BR with an existential variable is not horn. We in the following see that restricting to Horn BRs, size of the chase becomes polynomial w.r.t. size of the input quad-system, and the complexity of CCQ entailment further reduces compared to safe quad-systems.

Lemma 5.1. *For any Horn quad-system $QS_\mathcal{C} = \langle Q_\mathcal{C}, R\rangle$, the following holds: (i) $|chase(QS_\mathcal{C})| = \mathcal{O}(|QS_\mathcal{C}|^4)$ (ii) $chase(QS_\mathcal{C})$ can be computed in* EXPTIME *(iii) If $|R|$ is fixed to be a constant, $chase(QS_\mathcal{C})$ can be computed in* PTIME.

Proof. (i) Since each c, s, p, o, for any $c\colon (s,p,o) \in Q_\mathcal{C}$, is a constant, the number of constants in $QS_\mathcal{C}$, is given as $|\mathbf{C}(QS_\mathcal{C})| = \mathcal{O}(4 * |QS_\mathcal{C}|)$. As no blank node generating Skolem function occur in any BR in a Horn quad-system $QS_\mathcal{C}$, the set of constants $\mathbf{C}(chase(QS_\mathcal{C}))$ in its chase, is such that $\mathbf{C}(chase(QS_\mathcal{C})) = \mathbf{C}(QS_\mathcal{C})$. Since each $c\colon (s,p,o) \in chase(QS_\mathcal{C})$ is such that $c,s,p,o \in \mathbf{C}(QS_\mathcal{C})$, $|chase(QS_\mathcal{C})| = \mathcal{O}(|\mathbf{C}(QS_\mathcal{C})|^4) = \mathcal{O}(|QS_\mathcal{C}|^4)$.

(ii) Since from (i) $|chase(QS_\mathcal{C})| = \mathcal{O}(|QS_\mathcal{C}|^4)$, and in each iteration of the chase at least one new quad should be added, the number of iterations cannot exceed $\mathcal{O}(|QS_\mathcal{C}|^4)$. Since by lemma 3.1, each iteration i of chase computation requires $\mathcal{O}(|chase_{i-1}(QS_\mathcal{C})|^{|R|})$ time, and $|R| \leq |QS_\mathcal{C}|$, time required for each iteration is of the order $\mathcal{O}(2^{|QS_\mathcal{C}|})$ time. Since each iteration requires EXPTIME, although the number of iterations is a polynomial, total time required for chase computation is in EXPTIME.

(iii) As we know that the time taken for application of a BR R is $\mathcal{O}(|chase_{i-1}(QS_\mathcal{C})|^{|R|})$. Since $|R|$ is fixed to a constant, application of R can be done in PTIME. Also we know that $owl\text{-}horst\text{-}closure$ can be computed in PTIME. Hence, each chase iteration can be computed in PTIME. Also since the number of iterations is a polynomial in $|QS_\mathcal{C}|$, computing chase is in PTIME. □

Theorem 5.2. *Data complexity of CCQ entailment over Horn quad-systems is* PTIME*-complete.*

Proof. (Membership) Follows from the membership in P of data complexity of CCQ entailment for safe quad-systems that are more expressive than horn quad-systems (Theorem 4.9).

(Hardness) In order to prove PTIME-hardness, we reduce a well known PTIME-hard problem of "reachability of two nodes in a directed graph" which is a well known PTIME-complete problem. Given a graph $G = \langle V, E\rangle$, where $E \subseteq V \times V$, and any two nodes $s, t \in V$, to determine where t is reachable from s is a PTIME-hard problem. We reduce this problem to CCQ evaluation problem over a quad-system whose set of schema triples, the set of BRs, and the query CQ are all fixed. Given any graph $G = \langle V, E\rangle$, a source node s and a target node t. We create a quad-system $QS_c = \langle Q_c, \emptyset\rangle$, where instance set (corresponds to A-box or Data) of Q_c, $Abox(Q_c) = \{c\colon (v, edge, v') | (v, v') \in E\} \cup \{c\colon (s, \texttt{rdf:type}, A)\}$, the constant sized schema set (corresponds to T-box) of Q_c, i.e. $Tbox(Q_c) = \{A \sqsubseteq \forall edge.A\}$, which is an OWL Horst compliant axiom, as OWL-Horst allows universal restrictions on right hand side of sub-class expressions. Now it is easy to see that $QS_c \models c\colon (t, \texttt{rdf:type}, A)$ iff t is reachable from s. □

Theorem 5.3. *Combined complexity of CCQ entailment over a Horn quad-system is* EXPTIME-*complete.*

Proof. (Membership) By lemma 5.1, for any Horn quad-system $QS_\mathcal{C}$, its chase $chase(QS_\mathcal{C})$, can be computed in EXPTIME. Also by lemma 5.1, its chase size $|chase(QS_\mathcal{C})|$ is a polynomial w.r.t to $|QS_\mathcal{C}|$. Since a boolean CCQ $CQ()$ can naively be evaluated by grounding the set of constants in the chase to the variables in the $CQ()$, and then checking if any of these groundings are contained in $chase(QS_\mathcal{C})$. The number of such groundings can at most be $|chase(QS_\mathcal{C})|^{|CQ()|}$ (†). Since $|chase(QS_\mathcal{C})|$ is polynomial in $QS_\mathcal{C}$, there are an exponential number of groundings w.r.t $|CQ()|$. Since containment of each of these groundings can be checked in $chase(QS_\mathcal{C})$ in PTIME, as $|chase(QS_\mathcal{C})|$ is a polynomial w.r.t. $|QS_\mathcal{C}|$. Hence, the time complexity of CCQ entailment is in EXPTIME.

(Hardness) For EXPTIME-hardness, since we already saw in subsection 4.1 that with appropriate BRs and triple patterns one can simulate an alternating turing machine. The proof can slightly be modified to simulate an EXPTIME deterministic turing machine (DTM). The steps in the proof is same as the one in Dantsin et al. [23], where EXPTIME hardness of function-free Horn logic programs (Datalog) is shown. □

5.1. Restricted Horn Quad-Systems

We call those quad-systems with BRs of form (4) with a fixed bound on n as *restricted Horn quad-systems*. They can be further classified as linear, quadratic, cubic,..., quad-systems, when $n = 1, 2, 3, ...$, respectively.

Theorem 5.4. *Data complexity of CCQ entailment over restricted Horn quad-systems is* P-*complete.*

Proof. The proof is same as in theorem 5.2, since the size of BRs are fixed to constant. □

Theorem 5.5. *Combined complexity of CCQ entailment over restricted Horn quad-systems is* NP-*complete.*

Proof. Let the decision problem of determining if $QS_\mathcal{C} \models CQ()$ be called DP.

(Membership) for any $QS_\mathcal{C}$ whose rules are of restricted Horn-type, by lemma 5.1, its $chase(QS_\mathcal{C})$ can be computed in PTIME in the size of $QS_\mathcal{C}$ and $chase(QS_\mathcal{C})$ has a polynomial number of constants. Hence, if we guess an assignment μ for all the existential variables in CCQ $CQ()$, to the set of constants in $chase(QS_\mathcal{C})$. Then, one can evaluate the CCQ by checking if $c\colon (s, p, o) \in chase(QS_\mathcal{C})$, for each $c\colon (s, p, o) \in CQ()[\mu]$, which can be done in time $\mathcal{O}(|CQ| * |chase(QS_\mathcal{C})|)$, and is hence in PTIME. Hence, a machine that can make a non-deterministic guess can decide DP in polynomial time. Hence DP is in NP.

(Hardness) We show that DP is NP-hard, by reducing the well known NP-hard problem, 3-colorability, to DP. Given a graph $G = \langle V, E\rangle$, where $V = \{v_1, ..., v_n\}$ is the set of nodes, $E \subseteq V \times V$ is the set of edges, 3-colorability problem, is to decide if there exists a labeling function $l : V \rightarrow \{r, b, g\}$ that assigns each $v \in V$ to an element in $\{r, b, g\}$ such that the condition: $(v, v') \in E \rightarrow l(v) \neq l(v')$, for each $(v, v') \in E$, is satisfied.

One can construct a quad-system $QS_c = \langle Q_c, \emptyset\rangle$, where $graph_{Q_c}(c)$ has the following triples:

$\{(r, edge, b), (r, edge, g), (b, edge, g), (b, edge, r), (g, edge, r), (g, edge, b)\}$

Let CQ be the boolean CCQ: $\exists v_1, ..., v_n \bigwedge_{(v,v')\in E}$ $[\ c\colon (v, edge, v') \wedge c\colon (v', edge, v)]$. Then, it can be seen that G is 3-colorable, iff $QS_c \models CQ$. □

6. Related Work

Contexts and Distributed Logics The work on contexts began in the 80s when McCarthy [1] proposed context as a solution to the generality problem in AI. After this various studies about logics of contexts mainly in the field of KR was done by Guha [15], *Distributed First Order Logics* by Ghidini et al. [14] and *Local Model Semantics* by Giunchiglia et al. [9]. Primarily in these works contexts are formalized as a first order/propositional theory and bridge rules were provided to inter-operate the various theories of contexts. Some of the initial works on contexts relevant to semantic web were the ones like *Distributed Description Logics* [5] by Borgida et al., *E-connections* [19] by Kutz et al., and *Context-OWL* [8] Bouqet et al., and the recent work of CKR [11,10] by Serafini et al. These were mainly logics based on DLs, which formalized contexts as OWL KBs, whose semantics is given using a distributed interpretation structure with additional semantic conditions that suits varying requirements. Compared to these works, the bridge rules we consider are much more expressive with conjunctions and existential variables that supports value-creation or blank node creation. Also, none of the above works are focused on the query answering problem, which is main focus of our work.

Temporal RDF/Annotated RDF Studies in extending standard RDF with dimensions such as time and annotations has already been accomplished. Gutierrez et al. in [38] tried to add a temporal extension RDF and defines the notion of a 'temporal rdf graph', in which a triple is augmented to a quadruple of form $t\colon (s, p, o)$, where t is a time point. Whereas annotated extensions to RDF and querying annotated graphs has been studied in Udrea et al. [39] and Straccia et al. [40]. Unlike the case of time, here the quadruple has the form: $a\colon (s, p, o)$, where a is an annotation. The authors provide semantics, inference rules and query language that allows to express temporal/annotated queries. Although these approaches, in a way address contexts by means of time and annotations, the main difference in our work is that we provide the means to specify expressive bridge rules for inter-operating the reasoning between the various contexts.

Existential rules, TGDs, Datalog+- rules Query answering over rules with universal-existential quantifiers in the context of databases, where these rules are called Datalog+- rules/tuple generating dependencies (TGDs), was done by Beeri and Vardi [12] even in the early 80s, where the authors show that the query entailment problem in general is undecidable. However, recently many classes of such rules have been identified for which query answering is decidable. Some of these works that guarantees a finite chase is focused on techniques that detects 'acyclicity conditions' that guarantees chase termination by analyzing the information flow between rules have been proposed. *Weak acyclicity* [21,22], was one of the first such notions, and was extended to *joint acyclicity* [36] and *super weak acyclicity* [35]. The main approach used in these techniques is to exploit the structure of the rules and use a dependency graph that models the propagation path of constants across various predicates in the rules, and restricting the dependency graph to be acyclic. However, it is well known that these approaches produces a large number of false alarms, i.e. it is often the case that although dependency graph is cyclic, the chase is finite. Although these approaches can be employed in our scenario, if one translates a quad-system to a set of TGDs, this will inherit all the inherent drawbacks of the approaches based on dependency graphs. Hence, more recently, techniques other than the ones based on weak acyclicity has been proposed. These includes fragments of TGDs such that the resulting models have bounded tree widths by Baget et al. [13], Weakly guarded rules [6], and 'sticky' rules by cali et al. [34]. The approach used for query answering in these works is to rewrite the input query w.r.t. to the TGDs to another query that can be evaluated directly on the set of instances, such that the answers for the former query and latter query coincides. The approach is called the *query rewriting approach*. Also compared to our approach, for which the chase is finite, these approaches do not enjoy the finite chase property, and is hence not conducive to materialization/forward chaining based query answering.

Data integration Studies in query answering on integrated heterogeneous databases with expressive integration rules in the realm of data integration is primarily studied in the following two settings: (i) Data exchange [21], in which there is a source database and target database that are connected with existential rules, and (ii) Peer-to-peer data management systems (PDMS) [16], where there are an arbitrary number of peers that are interconnected using existential rules.

The approach based on dependency graph, for instance, is used by Halevi et al. in the context of peer-peer data management systems [16], and decidability is attained by not allowing any kind cycles in the peer topology. Whereas in the context of Data exchange, weak acyclicity is used in [21] to assure decidability, and the recent work by Marnette [35] employs the super weak acyclicity to ensure decidability. It can straightforwardly noted that our notion of safety is a generalization of these acyclicity based approaches. This is because when a quad-system is unsafe, requires skolem blank-node generated in a (set of) context(s) to be a sub-blank-node of another blank-node generated in the same set of context(s). This means that the former blank-node should propagate back to context(s) where it was generated, and hence needs cyclic dependency paths. Because of this, our approach can straightforwardly be employed in these systems.

DL+rules Works on extending DL KBs with Datalog like rules was studied by Horrocks et al.[27] giving rise to the SWRL[27] language. The related initiatives proposes a formalism using which one can mix a DL ontology with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language, and hence enables horn-like rules to be combined with an OWL KB. Since SWRL is undecidable in general, studies on computable sub-fragments gave rise to works like Description Logic Rules [37], where the authors deal with rules that can be totally internalized by a DL knowledge base, and hence if the DL considered is decidable, then also is a DL+rules KB. The authors give various fragments of the rule bases like SROIQ rules, EL++ rules etc. and show that certain new constructs that are not expressible by plain DL can be expressed using rules although they are finally internalized into DL KBs. Unlike in our scenario, these works consider only horn rules with out existential variables.

7. Summary and Conclusion

In this paper, we study the problem of query answering over contextualized RDF knowledge. We show that the problem in general is undecidable, and present few decidable classes of quad-systems. Table 1 displays the complexity results of chase computation and query entailment for the various fragments of quad-systems, we have derived. We can show that the notion of safety, introduced in section 4 can be used to extend the currently established tools for contextual reasoning to give support for expressive bridge rules with conjuction and existential quantifiers with decidability guarantees. We view the semantics and the results obtained in this paper as a general foundation for contextual reasoning and query answering over contextualized RDF knowledge formats such as Quads, and can straightforwardly be used to extend existing knowledge stores like Sesame/4store.

8. Acknowledgements

I sincerely thank my Advisor, Luciano Serafini (FBK-IRST, Trento), for all his valuable mentoring and guidance he has extended to me in these years. I also thank Prof. Gabriel Mark Kuper (DISI, University of Trento), Loris Bozzatto (FBK-IRST), and Francesco Corcoglionitti (FBK-IRST), for all their support and giving me all the helpful feedbacks that improved the quality of this work significantly. I also thank Prof. Roberto Zunino (DISI, Trento) who showed to me in a very productive discussion that a deterministic turing machine can be simulated by the unrestricted quad-systems.

References

[1] J.McCarthy, "Generality in AI," *Comm. of the ACM*, vol. 30, no. 12, pp. 1029–1035, 1987.

[2] J. Carroll, C. Bizer, P. Hayes, and P. Stickler, "Named graphs, provenance and trust," in *Proc. of the 14th int.l. conf. on WWW*, (New York, NY, USA), pp. 613–622, ACM, 2005.

[3] B. Schueler, S. Sizov, S. Staab, and D. T. Tran, "Querying for meta knowledge," in *WWW '08: Proceeding of the 17th international conference on World Wide Web*, (New York, NY, USA), pp. 625–634, ACM, 2008.

[4] D.Lenat, "The Dimensions of Context Space," tech. rep., CYCorp, 1998. Published online `http://www.cyc.com/doc/context-space.pdf`.

[5] A. Borgida and L. Serafini, "Distributed Description Logics: Assimilating Information from Peer Sources," *J. Data Semantics*, vol. 1, pp. 153–184, 2003.

[6] A. Cali, G. Gottlob, T. Lukasiewicz, B. Marnette, and A. Pieris, "Datalog+/-: A Family of Logical Knowledge Representation and Query Languages for New Applications," in *Logic in Computer Science (LICS), 2010 25th Annual IEEE Symposium on*, pp. 228 –242, july 2010.

[7] J. McCarthy, S. Buvac, T. Costello, R. Fikes, M. Genesereth, and F. Giunchiglia, "Formalizing Context (Expanded Notes)," 1995.

[8] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. C-OWL: Contextualizing Ontologies. In *ISWC*, pages 164–179, 2003.

[9] F. Giunchiglia and C. Ghidini. Local models semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 127, 2001.

[10] M.Joseph and L.Serafini. Simple reasoning for contextualized RDF knowledge. In *Proc. of Workshop on Modular Ontologies (WOMO-2011)*, 2011.

[11] L. Serafini and M. Homola. Contextualized knowledge repositories for the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2012.

[12] C. Beeri and M. Y. Vardi. The Implication Problem for Data Dependencies. In *ICALP*, pages 73–85, 1981.

[13] J.F. Baget, M.L. Mugnier, S. Rudolph, and M. Thomazo. Walking the Complexity Lines for Generalized Guarded Existential Rules. In *IJCAI*, pages 712–717, 2011.

[14] C. Ghidini and L. Serafini. Distributed first order logics. In *Frontiers Of Combining Systems 2, Studies in Logic and Computation*, pages 121–140. Research Studies Press, 1998.

[15] R.Guha. *Contexts: a Formalization and some Applications*. PhD thesis, Stanford, 1992.

[16] A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov, "Schema mediation in peer data management systems," in *In ICDE*, pp. 505–516, 2003.

[17] D. S. Johnson and A. C. Klug, "Testing containment of conjunctive queries under functional and inclusion dependencies," *J. Comput. Syst. Sci.*, vol. 28, no. 1, pp. 167–189, 1984.

[18] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*. Addison-Wesley, 1995.

[19] O. Kutz, C. Lutz, F. Wolter, and M. Zakharyaschev, "E-Connections of Abstract Description Systems," *Artificial Intelligence*, vol. 156, no. 1, pp. 1–73, 2004.

[20] S. Klarman and V. Gutiérrez-Basulto, "Two-dimensional description logics for context-based semantic interoperability," in *Proceedings of AAAI-11*, 2011.

[21] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa, "Data ex-

Quad-System Fragment	Chase size w.r.t input quad-system	Data Complexity of CCQ entailment	Combined Complexity of CCQ entailment
Unrestricted Quad-Systems	Infinite	Undecidable	Undecidable
Safe Quad-Systems	Triple exponential	PTIME-complete	3EXPTIME-complete
Horn Quad-Systems	Polynomial	PTIME-complete	EXPTIME-complete
Restricted Horn Quad-Systems	Polynomial	PTIME-complete	NP-complete

Table 1

Complexity info for various quad-system fragments

change: Semantics and query answering," in *In ICDT*, pp. 207–224, 2003.

[22] A. Deutsch and V. Tannen, "Reformulation of XML Queries and Constraints," in *In ICDT*, pp. 225–241, 2003.

[23] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *Computing Surveys (CSUR*, 33(3), September 2001.

[24] H. J. ter Horst, "Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary," *Web Semantics: Science, Services and Agents on the WWW*, vol. 3, no. 2-3, pp. 79–115, 2005. Selcted Papers from the ISWC, 2004.

[25] B. Marnette, "Generalized schema-mappings: from termination to tractability," in *Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '09, (New York, NY, USA), pp. 13–22, ACM, 2009.

[26] B. Glimm, C. Lutz, I. Horrocks, and U. Sattler, "Answering conjunctive queries in the $\mathcal{SHIQ}$ description logic," in *Proceedings of the IJCAI'07*, pp. 299–404, AAAI Press, 2007.

[27] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," w3c member submission, World Wide Web Consortium, 2004.

[28] A. Deutsch, A. Nash, and J. Remmel, "The chase revisited," in *Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '08, (New York, NY, USA), pp. 149–158, ACM, 2008.

[29] P. Hayes, ed., *RDF Semantics*. W3C Recommendation, Feb. 2004.

[30] B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz, "OWL 2 Web Ontology Language – Profiles," tech. rep., W3C, 2009.

[31] M. Krötzsch, "The not-so-easy task of computing class subsumptions in OWL RL," in *Proceedings of the 11th International Semantic Web Conference*, LNCS, Springer, 2012.

[32] M. A. Harrison, *Introduction to Formal Language Theory*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1st ed., 1978.

[33] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer, "Alternation," *J. ACM*, vol. 28, pp. 114–133, Jan. 1981.

[34] A. Calì, G. Gottlob, and A. Pieris, "Query Answering under Non-guarded Rules in Datalog+/-," in *RR* (P. Hitzler and T. Lukasiewicz, eds.), vol. 6333 of *Lecture Notes in Computer Science*, pp. 1–17, Springer, 2010.

[35] B. Marnette, "Generalized schema-mappings: from termination to tractability," in *Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '09, (New York, NY, USA), pp. 13–22, ACM, 2009.

[36] M. Krötzsch and S. Rudolph, "Extending decidable existential rules by joining acyclicity and guardedness," in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)* (T. Walsh, ed.), pp. 963–968, AAAI Press/IJCAI, 2011.

[37] M. Krötzsch, S. Rudolph, and Pascal Hitzler. Description logic rules. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI'08)*, pages 80–84. IOS Press, 2008.

[38] C. Gutierrez, C. A. Hurtado, and A. A. Vaisman, "Temporal rdf," in *ESWC*, pp. 93–107, 2005.

[39] O. Udrea, D. R. Recupero, and V. S. Subrahmanian, "Annotated rdf," *ACM Transactions in Computational Logic*, vol. 11, no. 2, pp. 1–41, 2010.

[40] U. Straccia, N. Lopes, G. Lukacsy, and A. Polleres, "A general framework for representing and reasoning with annotated semantic web data," in *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010), Special Track on Artificial Intelligence and the Web*, July 2010.

[41] P. F. Patel-Schneider, P. Hayes, and I. Horrocks, "OWL Web Ontology Language Semantics and Abstract Syntax Section 5. RDF-Compatible Model-Theoretic Semantics," tech. rep., W3C, Dec. 2004.

Appendix

A. RDF and Tractable OWL Extensions: Overview

Let $\mathbf{U}$ be the set of URIs, $\mathbf{B}$ the set of blank nodes and $\mathbf{L}$ the set of literals. The set $\mathbf{C} = \mathbf{U} \cup \mathbf{B} \cup \mathbf{L}$ are called the set of (RDF) constants. Any $(s, p, o) \in \mathbf{C} \times \mathbf{C} \times \mathbf{C}$ is called a generalized RDF triple (from now on, just triple). A graph is defined as a set of triples. RDF(S) [29], and OWL-Horst [24] are popular languages with semantics that gives special meaning to the terms in their vocabularies, and can be used for reasoning over graphs. By logical expressivity, these languages are grouped into the following hierarchy: RDF $\subseteq$ RDFS $\subseteq$ OWL-Horst. OWL-Horst, is a semantic extension to RDFS that defines a set of semantic conditions to a subset of terms in the OWL vocabulary. These include class assertions such as OWL restrictions (universal, existential, value restrictions), disjointness of classes and properties, property assertions like symmetricity, transitivity, functionality, inverse relations of properties and assertions involving `owl:sameAs` and `owl:differentFrom`. Like RDF(S), any ontology serialized as a graph can be reasoned using the OWL-Horst semantics. An OWL-Horst interpretation structure is a tuple ⟨IR, IP, IC, IEXT, ICEXT, IS, LV⟩, where IR is the object domain, IP $\subseteq$ IR is the property domain, IC $\subseteq$ IR is the class domain, IEXT, the property extension function, ICEXT, the class extension function, IS, the term interpretation function, and LV $\subseteq$ IR, are the set of literal values, a list of with additional semantic restrictions [24]. The class of OWL-Horst interpretation structures are a subset of the class of RDFS interpretation structures, as an OWL-Horst interpretation structure is an extension of a standard RDFS interpretation structure with additional semantic restrictions. OWL-Horst has a set of inference rules that are sound and complete w.r.t. its semantics, such that for any OWL-Horst graph g, its deductive closure, $owl\text{-}horst\text{-}closure(g)$, can be computed by repeatedly running the set of OWL-Horst inference rules on g until a fixpoint is reached. OWL-Horst reasoning for a graph can be characterized with the help of an OWL-Horst *canonical model*, which is an OWL-Horst model that represents all the OWL-Horst models of a graph, and is defined as:

Definition A.1 (OWL-Horst Canonical Model)**.** *For any OWL-Horst graph g, its canonical model* $can_{owl\text{-}horst}(g) = \langle IR^{c(g)}, IP^{c(g)}, IC^{c(g)}, IEXT^{c(g)}, ICEXT^{c(g)}, IS^{c(g)}, LV^{c(g)}\rangle$ *is an OWL-Horst interpretation structure, constructed as follows:*

- $LV^{c(g)} = \{l | l$ *is a plain literal and* l *occurs in* $owl\text{-}horst\text{-}closure(g)\} \cup \{dv(l) | l$ *is a datatyped literal occuring in* $owl\text{-}horst\text{-}closure(g)$, *where* $dv(l)$ *is the data value of* l $\}$
- $IP^{c(g)} = \{P | (P, \texttt{rdf:type}, \texttt{rdf:Property}) \in owl\text{-}horst\text{-}closure(g)\}$
- $IC^{c(g)} = \{C | (C, \texttt{rdf:type}, \texttt{rdfs:Class}) \in owl\text{-}horst\text{-}closure(g)\}$
- $IR^{c(g)} = LV^{c(g)} \cup IP^{c(g)} \cup IC^{c(g)} \cup \{a | (a, \texttt{rdf:type}, \texttt{rdfs:Resource}) \in owl\text{-}horst\text{-}closure(g)\}$
- $IS^{c(g)} = \{(a, a) | a$ *is any URI, blank node, or Plain literal that occurs in* $owl\text{-}horst\text{-}closure(g)\} \cup (l, dv(l)) | l$ *is a datatyped literal occuring in* $owl\text{-}horst\text{-}closure(g)$, *where* $dv(l)$ *is the data value of* l $\}$
- *for every* $P \in IP^{c(g)}, IEXT^{c(g)}(P) = \{(s, o) | (s, P, o) \in owl\text{-}horst\text{-}closure(g)\}$
- *for every* $C \in IC^{c(g)}, ICEXT^{c(g)}(C) = \{a | (a, \texttt{rdf:type}, C) \in owl\text{-}horst\text{-}closure(g)\}$

Consistency, as defined in [24] for an OWL-Horst graph, determines if the graph have any clashes or not. A clash, denoted by the symbol FALSE can result from invalid datatyped literals, and also from the presence of statements like $(a, \texttt{owl:sameAs}, b)$ and $(a, \texttt{owl:differentFrom}, b)$. Any graph, g, is said to be *OWL-Horst inconsistent*, if $g \models_{\text{owl-horst}}$ FALSE, and otherwise said to be *OWL-Horst consistent*, where $\models_{\text{owl-horst}}$ is the OWL-Horst entailment relation. We denote by $\vdash_{\text{owl-horst}}$ the derivability relation between a graph and a (set of) triple(s) using OWL-Horst inference rules. For any two OWL-Horst consistent graphs g, h, the following are true:

- $can_{\text{owl-horst}}(g) \models_{\text{owl-horst}} g$
- $can_{\text{owl-horst}}(g)$ can be computed in PTIME
- $g \models_{\text{owl-horst}} h$ iff $can_{\text{owl-horst}}(g) \models_{\text{owl-horst}} h$.

The proofs of the above facts can be found in [24]. *OWL 2 RL RDF rules [30]* is a partial axiomatization of OWL 2 RDF based semantics. These set of rules provides axiomatizations for OWL constructs like `owl:intersectionOf`, `owl:unionOf`, `owl:complementOf` which are not provided by OWL-Horst. Although deductive closure w.r.t these rules for any graph g can be computed in PTIME, the set of rules are incomplete for the OWL 2 RL fragment of OWL for reasoning tasks such as computing subsumptions, which is co-NP Hard [31].

B. Proofs of Section 3

Lemma 3.1. Let $r \in R$ be a BR, such that for any other $r' \in R$, $|r'| \leq |r|$, i.e. r is the BR in R, with the highest number of quad-patterns, and let $l = |r|$. (i) r can be applied on $chase_{i-1}(QS_{\mathcal{C}})$ by grounding variables in r to the set of constants in $chase_{i-1}(QS_{\mathcal{C}})$, the number of such groundings is of the order $\mathcal{O}(|chase_{i-1}(QS_{\mathcal{C}})|^l)$. Hence, $|r(chase_{i-1}(QS_{\mathcal{C}}))| = \mathcal{O}(l * |chase_{i-1}(QS_{\mathcal{C}})|^l)$ and $|R(chase_{i-1}(QS_{\mathcal{C}})| = \mathcal{O}(R * |chase_{i-1}(QS_{\mathcal{C}})|^l)$. Since $chase_i(QS_{\mathcal{C}}) = owl\text{-}horst\text{-}closure(S)$, where $S = chase_{i-1}(QS_{\mathcal{C}}) \cup R(chase_{i-1}(QS_{\mathcal{C}}))$. Since $|S| = \mathcal{O}(R * |chase_{i-1}(QS_{\mathcal{C}})|^l)$, and each member of the set S is a quad, the number of constants in $S, \mathbf{C}(S) = 4 * \mathcal{O}(R * |chase_{i-1}(QS_{\mathcal{C}})|^l)$. Since each s, p, o, such that $c\colon (s, p, o) \in owl\text{-}horst\text{-}closure(S)$ is from the set $\mathbf{C}(S)$,

$|chase_i(QS_{\mathcal{C}})| \leq |\mathcal{C}| * 64 * R^3 * |chase_{i-1}(QS_{\mathcal{C}})|^{3l}$. Since $|\mathcal{C}| \leq |QS_{\mathcal{C}}|$, $|chase_i(QS_{\mathcal{C}})| = \mathcal{O}(|QS_{\mathcal{C}}|^4 * |chase_{i-1}(QS_{\mathcal{C}})|^l$. Since $l \leq |R|$, $|chase_i(QS_{\mathcal{C}})|$ is of the order $\mathcal{O}(|QS_{\mathcal{C}}| * |chase_{i-1}(QS_{\mathcal{C}})|^{|R|}$.

(ii) From (i) we know that $|R(chase_{i-1}(QS_{\mathcal{C}}))| = \mathcal{O}(R * |chase_{i-1}(QS_{\mathcal{C}})|^l)$. Since, no new constant is introduced in any of the non-generating iterations $i+1, ..., i+j$, the set of constants in iteration $i+j$ is: $\mathbf{C}(chase_{i+j}(QS_{\mathcal{C}})) = \mathbf{C}(S)$, where $S = R(chase_{i-1}(QS_{\mathcal{C}})) \cup chase_{i-1}(QS_{\mathcal{C}})$. Since, we already saw in (i) that $|\mathbf{C}(S)| = 4 * \mathcal{O}(R * |chase_{i-1}(QS_{\mathcal{C}})|^l)$, $|\mathbf{C}(chase_{i+j}(QS_{\mathcal{C}}))| = 4 * \mathcal{O}(R * |chase_{i-1}(QS_{\mathcal{C}})|^l)$. Since each s, p, o, such that $c\colon (s,p,o) \in chase_{i+j}(QS_{\mathcal{C}})$ is from the set $\mathbf{C}(chase_{i+j}(QS_{\mathcal{C}}))$, $|chase_{i+j}(QS_{\mathcal{C}})| \leq |\mathcal{C}| * 64 * R^3 * |chase_{i-1}(QS_{\mathcal{C}})|^{3l}$. Since $|\mathcal{C}| \leq |QS_{\mathcal{C}}|$, $|chase_{i+j}(QS_{\mathcal{C}})| = \mathcal{O}(|QS_{\mathcal{C}}|^4 * |chase_{i-1}(QS_{\mathcal{C}})|^l)$. Since $l \leq |R|$, $|chase_i(QS_{\mathcal{C}})|$ is of the order $\mathcal{O}(|QS_{\mathcal{C}}| * |chase_{i-1}(QS_{\mathcal{C}})|^{|R|})$. ☐

Proposition B.1. *There exists unrestricted quad-systems whose dChase is infinite.*

Proof. Consider an example of a quad-system $QS_c = \langle Q_c, r\rangle$, where $graph_{Q_c}(c) = \{a, \texttt{rdf:type}, C\}$, and the BR $r = c1\colon (x, \texttt{rdf:type}, C) \to c_1\colon (x, P, f(x)), c1\colon (f(x), \texttt{rdf:type}, C)$. The dChase computation starts with $dChase_0(QS_c) = \{c\colon (a, \texttt{rdf:type}, C)\}$, now the rule r is applicable, and its application leads to $dChase_1(QS_c) = \{c\colon (a, \texttt{rdf:type}, C), c\colon (a, P, f(a)), c\colon (f(a), \texttt{rdf:type}, C)\}$, which again is applicable by r for $c\colon (f(a), \texttt{rdf:type}, C)$. For any $i \geq 0, dChase_i(QS_c)$ contains $c\colon (f^i(a), \texttt{rdf:type}, C)$ which in turn is applicable by r. Hence $dChase_i(QS_c)$ does not have a finite fix-point, and hence $dChase(QS_c)$ is infinite. ☐

Theorem 3.2. We show that CCQ entailment is undecidable for unrestricted quad-systems, by showing that the well known undecidable problem of "non-emptiness of intersection of context-free grammars" is reducible to the CCQ answering problem.

Given an alphabet Σ, string $\vec{w}$ is a sequence of symbols from Σ. A language L is a subset of Σ^*, where Σ^* is the set of all strings that can be constructed from the alphabet Σ, and also includes the empty string ϵ. Grammars are machineries that generate a particular language. A grammar G is a quadruple $\langle V, T, S, P\rangle$, where V is the set of variables, T, the set of terminals, $S \in V$ is the start symbol, and P is a set of production rules (PR), in which each PR $r \in P$, is of the form:

$$\vec{w} \to \vec{w}'$$

where $\vec{w}, \vec{w}' \in \{T \cup V\}^*$. Intuitively application of a PR r of the form above on a string $\vec{w}_1$, replaces every occurrence of the sequence $\vec{w}$ in $\vec{w}_1$ with $\vec{w}'$. PRs are applied starting from the start symbol S until it results in a string $\vec{w}$, with $\vec{w} \in \Sigma^*$ or no more production rules can be applied on $\vec{w}$. In the former case, we say that $\vec{w} \in L(G)$, the language generated by grammar G. For a detailed review of grammars, we refer the reader to Harrison et al. [32]. A *context-free grammar* (CFG) is a grammar, whose set of PRs P, have the following property:

Property B.2. *For a CFG, every PR is of the form $v \to \vec{w}$, where $v \in V$, $\vec{w} \in \{T \cup V\}^*$.*

Given two CFGs, $G_1 = \langle V_1, T, S_1, P_1\rangle$ and $G_2 = \langle V_2, T, S_2, P_2\rangle$, where V_1, V_2 are the set of variables, T such that $T \cap (V_1 \cup V_2) = \emptyset$ is the set of terminals. $S_1 \in V_1$ is the start symbol of G_1, and P_1 are the set of PRs of the form $v \to \vec{w}$, where $v \in V$, $\vec{w}$ is a sequence of the form $w_1 ... w_n$, where $w_i \in V_1 \cup T$. Similarly s_2, P_2 is defined. Deciding whether the language generated by the grammars $L(G_1)$ and $L(G_2)$ have non-empty intersection is known to be undecidable [32].

Given two CFGs, $G_1 = \langle V_1, T, S_1, P_1\rangle$ and $G_2 = \langle V_2, T, S_2, P_2\rangle$, we encode grammars G_1, G_2 into a quad-system of the form $QS_c = \langle Q_c, R\rangle$, with a single context identifier c. Each PR $r = v \to \vec{w} \in P_1 \cup P_2$, with $\vec{w} = w_1 w_2 w_3 .. w_n$, is encoded as a BR of the form:

$$c\colon (x_1, w_1, x_2), c\colon (x_2, w_2, x_3), ..., c\colon (x_n, w_n, x_{n+1}) \to c\colon (x_1, v, x_{n+1}) \tag{5}$$

where $x_1, .., x_{n+1}$ are variables. W.l.o.g. we assume that the set of terminal symbols T is equal to the set of terminal symbols occurring in $P_1 \cup P_2$. For each terminal symbol $t_i \in T$, R contains a BR of the form:

$$c\colon (x, \texttt{rdf:type}, C) \to \exists y\, c\colon (x, t_i, y), c\colon (y, \texttt{rdf:type}, C) \tag{6}$$

and Q_c contains only the triple:

$$c\colon (a, \texttt{rdf:type}, C)$$

We in the following show that:

$$QS_c \models \exists y\, c\colon (a, S_1, y) \wedge c\colon (a, S_2, y) \leftrightarrow L(G_1) \cap L(G_2) \neq \emptyset \qquad (7)$$

Claim (1) For any $\vec{w} = t_1, ..., t_p \in T^*$, there exists $b_1, ...b_p$, such that $c\colon (a, t_1, b_1)$, $c\colon (b_1, t_2, b_2)$, ..., $c\colon (b_{p-1}, t_p, b_p)$, $c\colon (b_p, \texttt{rdf:type}, C) \in dChase($ $QS_c)$.

we proceed by induction on the $|\vec{w}|$.

base case suppose if $|\vec{w}| = 1$, then $\vec{w} = t_i$, for some $t_i \in T$. But Since by construction $c\colon (a,$ $\texttt{rdf:type},\ C) \in dChase_0(QS_c)$, on which rules of the form (6) is applicable. Hence, there exists an i such that $dChase_i(QS_c)$ contains $c\colon (a, t_i, b_i)$, $c\colon (b_i, \texttt{rdf:type}, C)$, for each $t_i \in T$. Hence, the base case.

hypothesis for any $\vec{w} = t_1...t_p$, if $|\vec{w}| \leq p'$, then there exists $b_1, ..., b_p$, such that $c\colon (a, t_1, b_1)$, $c\colon (b_1, t_2, b_2)$, ..., $c\colon (b_{p-1}, t_p, b_p)$, $c\colon (b_p,$ $\texttt{rdf:type}, C) \in dChase(QS_c)$.

inductive step suppose $\vec{w} = t_1...t_{p+1}$, with $|\vec{w}| \leq p' + 1$. Since $\vec{w}$ can be written as $\vec{w'}t_{p+1}$, where $\vec{w}' = t_1...t_p$, and by hypothesis, there exists $b_1, ..., b_p$ such that $c\colon (a, t_1, b_1)$, $c\colon (b_1, t_2, b_2)$, ..., $c\colon (b_{p-1}, t_p, b_p)$, $c\colon (b_p, \texttt{rdf:type}, C) \in dChase(QS_c)$. Also since rules of the form (6) are applicable on $c\colon (b_p,\ \texttt{rdf:type},\ C)$, and hence produces triples of the form $c\colon (b_p, t_i, b^i_{p+1})$, $c\colon (b^i_{p+1}),\ \texttt{rdf:type},\ C)$, for each $t_i \in T$. Since $t_{p+1} \in T$, the claim follows.

For a grammar $G = \langle V, T, S, P \rangle$, whose start symbol is S, and for any $\vec{w} \in \{V \cup T\}^*$, for some $V_j \in V$, we denote by $V_j \rightarrow^i \vec{w}$, the fact that $\vec{w}$ was derived from V_j by i production steps, i.e. there exists steps $V_j \rightarrow r_1, ..., r_i \rightarrow \vec{w}$, which lead to the production of $\vec{w}$. For any $\vec{w}$, $\vec{w} \in L(G)$, iff there exists an i such that $S \rightarrow^i \vec{w}$. For any $V_j \in V$, we use $V_j \rightarrow^* \vec{w}$ to denote the fact that there exists an arbitrary i, such that $V_j \rightarrow^i \vec{w}$.

Claim (2) For any $\vec{w} = t_1...t_p \in \{V \cup T\}^*$, and for any $V_j \in V$, if $V_j \rightarrow^* \vec{w}$ and there exists $b_1, ..., b_{p+1}$, with $c\colon (b_1, t_1, b_2), ..., c\colon (b_p, t_p, b_{p+1}) \in dChase(QS_c)$, then $c\colon (b_1, V_j, b_{p+1}) \in dChase(QS_c)$.

We prove this by induction on the size of $\vec{w}$.

base case Suppose $|\vec{w}| = 1$, then $\vec{w} = t_k$, for some $t_k \in T$. If there exists b_1, b_2 such that $c\colon (b_1, t_k, b_2)$. But since there exists a PR $V_j \rightarrow t_k$, by transformation given in (5), there exists a BR $c\colon (x_1, t_k, x_2) \rightarrow c\colon (x_1, V_j, x_2) \in R$, which is applicable on $c\colon (b_1, t_k, b_2)$ and hence the quad $c\colon (b_1, V_j, b_2) \in dChase(QS_c)$.

hypothesis For any $\vec{w} = t_1...t_p$, with $|\vec{w}| \leq p'$, and for any $V_j \in V$, if $V_j \rightarrow^* \vec{w}$ and there exists $b_1, ...b_p, b_{p+1}$, such that $c\colon (b_1, t_1, b_2)$, ..., $c\colon (b_p, t_p, b_{p+1}) \in dChase(QS_c)$, then $c\colon (b_1,$ $V_j, b_{p+1}) \in dChase(QS_c)$.

inductive step Suppose if $\vec{w} = t_1...t_{p+1}$, with $|\vec{w}| \leq p' + 1$, and $V_j \rightarrow^i \vec{w}$, and there exists $b_1, ...b_{p+1}$, b_{p+2}, such that $c\colon (b_1, t_1, b_2)$, ..., $c\colon (b_{p+1}, t_{p+1},$ $b_{p+2}) \in dChase(Q_c)$. Also, one of the following holds (i) $i = 1$, or (ii) $i > 1$. Suppose (i) is the case, then it is trivially the case that $c\colon (b_1, V_j, b_{p+2}) \in dChase(QS_c)$. Suppose if (ii) is the case, one of the two sub cases holds (a) $V_j \rightarrow^{i-1} V_k$, for some $V_k \in V$ and $V_k \rightarrow^1 \vec{w}$ or (b) there exist a $V_k \in V$, such that $V_k \rightarrow^* t_{q+1}...t_{q+l}$, with $2 \leq l \leq p$, where $V_j \rightarrow^* t_1...t_q V_k t_{p-l+1}...t_{p+1}$. If (a) is the case, trivially then $c\colon (b_1, V_k, b_{q+2}) \in dChase(QS_c)$, and since by construction there exists $c\colon (x_0, V_k, x_1)$ $\rightarrow$ $c\colon (x_0, V_{k+1}, x_1)$, ..., $c\colon (x_0, V_{k+i}, x_1) \rightarrow c\colon (x_0, V_j, x_1) \in R$, $c\colon (b_1, V_j, b_{q+2}) \in dChase($ $QS_c)$. If (b) is the case, then since $|t_{q+1}...t_{q+l}| \geq 2$, $|t_1...t_q V_2 t_{p-l+1}...t_{p+1}| \leq p'$. This implies that $c\colon (b_1, V_j, b_{p+2}) \in dChase(QS_c)$.

Similarly, by construction of $dChase(QS_c)$, the following claim can straightforwardly be shown to hold:

Claim (3) For any $\vec{w} = t_1...t_p \in \{V \cup T\}^*$, and for any $V_j \in V$, if there exists $b_1, ..., b_p, b_{p+1}$, with $c\colon (b_1, t_1, b_2)$, ..., $c\colon (b_p, t_p, b_{p+1}) \in dChase(QS_c)$ and $c\colon (b_1, V_j, b_{p+1}) \in dChase(QS_c)$, then $V_j \rightarrow^* \vec{w}$.

(a) For any $\vec{w} = t_1...t_p \in T^*$, if $\vec{w} \in L(G_1) \cap L(G_2)$, then by claim 1, since there exists $b_1, ..., b_p$, such that $c\colon (a, t_1, b_1), ..., c\colon (b_{p-1}, t_p, b_p) \in dChase(QS_c)$. But since $\vec{w} \in L(G_1)$ and $\vec{w} \in L(G_2)$, $S_1 \rightarrow \vec{w}$ and $S_2 \rightarrow \vec{w}$. Hence by claim 2, $c\colon (a, S_1, b_p)$, $c\colon (a, S_2, b_p)$ $\in dChase(QS_c)$, which implies that $dChase(QS_c)$ $\models \exists y\ c\colon (a, s_1, y) \wedge c\colon (a, s_2, y)$. Hence, $QS_c \models \exists y$ $c\colon (a, s_1, y) \wedge c\colon (a, s_2, y)$.

(b) Suppose if $QS_c \models \exists y\, c\colon (a, S_1, y) \wedge c\colon (a, S_2, y)$, then this implies that there exists b_p such that $c\colon (a,$ $S_1,\ b_p)$, $c\colon (a, S_2, b_p) \in dChase(QS_C)$. Then it is the case that there exists $\vec{w} = t_1...t_p \in T^*$, and

$b_1, ..., b_p$ such that $c\colon (a, t_1, b_1)$, ..., $c\colon (b_{p-1}, t_p, b_p)$, $c\colon (a, S_1, b_p)$, $c\colon (a, S_2, b_p) \in dChase(QS_c)$. Then by claim 3, $S_1 \rightarrow^* \vec{w}$, $S_2 \rightarrow^* \vec{w}$. Hence, $w \in L(G_1) \cap L(G_2)$.

By (a),(b) it follows that there exists $\vec{w} \in L(G_1) \cap L(G_2)$ iff $QS_c \models \exists y\ c\colon (a, s_1, y) \wedge c\colon (a, s_2, y)$. As we have shown that the intersection of CFGs, which is an undecidable problem, is reducible to the problem of query entailment on unrestricted quad-system, the latter is undecidable. □

C. Proofs of Section 4

Lemma 4.7. (i) Any $c\colon (s, p, o) \in dChase(QS_\mathcal{C})$ is such that $s, p, o \in \mathbf{C}(dChase(QS_\mathcal{C}))$. Also $\mathbf{C}(dChase(QS_\mathcal{C})) = \mathbf{U}(dChase(QS_\mathcal{C})) \cup \mathbf{B}(dChase(QS_\mathcal{C})) \cup \mathbf{L}(dChase(QS_\mathcal{C}))$, and $\mathbf{U}(dChase(QS_\mathcal{C})) = \mathbf{U}(QS_\mathcal{C})$, $\mathbf{L}(dChase(QS_\mathcal{C})) = \mathbf{L}(QS_\mathcal{C})$. Also $\mathbf{B}(dChase(QS_\mathcal{C})) = \mathbf{B}(QS_\mathcal{C}) \cup \mathbf{B}_{sk}(dChase(QS_\mathcal{C}))$. Since $\mathbf{C}(chase(QS_\mathcal{C})$, $\mathbf{L}(chase(QS_\mathcal{C}))$, and $\mathbf{B}(QS_\mathcal{C})$ are part of the input, the set $\mathbf{B}_{sk}(dChase(QS_\mathcal{C}))$ determines the incremental part of the $chase(QS_\mathcal{C})$. Note that each $b \in \mathbf{B}_{sk}(dChase(QS_\mathcal{C}))$ is a skolem blank node, whose descendance graph can be unraveled into a tree that satisfies the set of constraints given in property 4.6. Since every non-leaf node from a path from the root to the leaf node has a distinct set of origin Contexts, the depth of any such tree is bounded by $2^{|\mathcal{C}|}$. Also since order of the tree is bounded by $m = max\{ar(f_i) | f_i$ is a skolem function symbol occurring in $sk(R)\}$, any such tree has at-most $m^{2^{|\mathcal{C}|}}$ leaf nodes and $m^{2^{|\mathcal{C}|}}$ non-leaf nodes. Let F be the set of function skolem symbols occurring in R. Since each leaf node is a constant in $\mathbf{C}(QS_\mathcal{C})$, and each non-leaf node is an element in F, the number of possible descendance trees is bounded by $|F|^{m^{2^{|\mathcal{C}|}}} * |\mathbf{C}(QS_\mathcal{C})|^{m^{2^{|\mathcal{C}|}}}$, which is triple exponential in $|QS_\mathcal{C}|$ as $|F|, m, |\mathcal{C}|, |\mathbf{C}(QS_\mathcal{C})|$ are polynomially bounded by fixed input size $|QS_\mathcal{C}|$. Hence, the number of skolem blank nodes $\mathbf{B}_{sk}(dChase(QS_\mathcal{C}))$ are finitely bounded by $\mathcal{O}(2^{2^{2^{|QS_\mathcal{C}|}}})$. Hence, $|\mathbf{C}(dChase(QS_\mathcal{C})|$ is bounded by $\mathcal{O}(2^{2^{2^{|QS_\mathcal{C}|}}})$, and $|dChase(QS_\mathcal{C})| = \mathcal{O}(2^{2^{2^{|QS_\mathcal{C}|}}})$.

(ii) From (i) $|dChase(QS_\mathcal{C})|$ is triply exponential in $|QS_\mathcal{C}|$, and since each iteration add at-least one quad to its dChase, the number of iterations are bounded triple exponentially in $|QS_\mathcal{C}|$. Also, by lemma 3.1 any iteration i can be done in time $\mathcal{O}(|dChase_{i-1}(QS_\mathcal{C})|^{|R|})$. Since, using (i) $|dChase_{i-1}(QS_\mathcal{C})| = \mathcal{O}(2^{2^{2^{|QS_\mathcal{C}|}}})$, each iteration i can be done in time $\mathcal{O}(2^{|R| * 2^{2^{|QS_\mathcal{C}|}}})$. Also, as number of iterations is triple exponential, computing $dChase(QS_\mathcal{C})$ is in 3EXPTIME.

(iii) Since $|R|$ is fixed to a constant, the set of skolem function symbols F in $sk(R)$, the arity of any $f \in F$, and set of origin contexts are constants. Because of this, the number of tree structures of skolem blank-nodes generated is a constant z. Hence, the number of inner nodes and leaves of any such tree, which can be taken by any constant in $\mathbf{C}(QS_\mathcal{C})$. Hence, the number of skolem blank nodes generated is $\mathcal{O}(|\mathbf{C}(QS_\mathcal{C})|^z)$. Hence, the set of constants in $dChase(QS_\mathcal{C})$ is a polynomial in $|QS_\mathcal{C}|$, and also is $|dChase(QS_\mathcal{C})|$.

Also, since in any dChase iteration except the final one, atleast one quad should be produced and the final dChase can have atmost $\mathcal{O}(|QS_\mathcal{C}|^z)$ triples, the total number of iterations are bounded by $\mathcal{O}(|QS_\mathcal{C}|^z)$ (†). Since, any dChase iteration i involves only the following two operations (a) $owl\text{-}horst\text{-}closure$ and (b) computing $R(dChase_{i-1}(QS_\mathcal{C}))$. (a) can be done in time polynomial w.r.t. its input [24]. Since, we already saw in (ii) that the time required for (b) is given by $|dChase_{i-1}(QS_\mathcal{C})|^{|R|}$, and since $|R|$ is a constant, this time required for (b) is a polynomial in the size its input. Hence, any dChase iteration can be performed in polynomial time w.r.t. its input (‡). From (†) and (‡), it can be concluded that dChase can be computed in PTIME. □

Lemma C.1 (Soundness). *For any quad-system, $QS_\mathcal{C} = \langle Q_\mathcal{C}, R\rangle$, if the constant **unSafe** $\in dChase^{safe}(QS_\mathcal{C})$, then $QS_\mathcal{C}$ is unsafe.*

Proof. In order to prove the theorem, we first prove a few supporting claims. The following claim shows that any triple $c\colon (s, p, o)$ with $c \in \mathcal{C}$ is derived in safe dChase, is also derived in its standard dChase. In this way, safe dChase do not generate any unsound triples in any context $c \in \mathcal{C}$.

Claim (1) For any quad $c\colon (s, p, o)$, where $c \in \mathcal{C}$, if $c\colon (s, p, o) \in dChase^{\text{safe}}(QS_\mathcal{C})$, then $c\colon (s, p, o) \in dChase(QS_\mathcal{C})$.

We prove this claim by induction on the number of iterations of $dChase_i(QS_\mathcal{C})$ and $dChase_i^{\text{safe}}(QS_\mathcal{C})$.

base case $i = 0$, trivially holds, since by construction $graph_{dChase_0(QS_\mathcal{C})}(c) = graph_{dChase_0^{\text{safe}}(QS_\mathcal{C})}(c)$, for any $c \in \mathcal{C}$.

hypothesis for any $i \leq k$, $c \in \mathcal{C}$, if $c\colon (s,p,o) \in dChase_i^{\text{safe}}(QS_{\mathcal{C}})$, then $c\colon (s,p,o) \in dChase_i(QS_{\mathcal{C}})$.

inductive step If $c\colon (s,p,o) \in dChase_{k+1}^{\text{safe}}(QS_{\mathcal{C}})$ and $c \in \mathcal{C}$, then either (i) $c\colon (s,p,o) \in dChase_k^{\text{safe}}(QS_{\mathcal{C}})$ or (ii) $c\colon (s,p,o) \notin dChase_k^{\text{safe}}(QS_{\mathcal{C}})$. If (i) is the case, then by our hypothesis $c\colon (s,p,o) \in dChase_k(QS_{\mathcal{C}})$, and by construction of dChase, $c\colon (s,p,o) \in dChase_{k+1}(QS_{\mathcal{C}})$. Else if (ii) is the case, then by construction of the safe dChase, there exists a quad-graph S, such that for any $c'\colon (s',p',o') \in S$, either (a) $c'\colon (s',p',o')$ is in $dChase_k^{\text{safe}}(QS_{\mathcal{C}})$ or (b) $c'\colon (s',p',o')$ is obtained by application of some $r \in aug(R)$ from $dChase_k^{\text{safe}}(QS_{\mathcal{C}})$, and $S \vdash_{horst} c\colon (s,p,o)$. If (a) is the case, then by hypothesis $c'\colon (s',p',o') \in dChase_{k+1}(QS_{\mathcal{C}})$, and if (b) is the case, then then there exists $\mu \in M$, and $r = body(r) \rightarrow head(r) \in aug(R)$, such that $body(r)[\mu] \in dChase_k^{\text{safe}}$, and $c'\colon (s',p',o') \in head(r)[\mu]$. By construction, for any $r \in aug(R)$, there exists $r' \in R$, such that $r = aug(r')$. Since, $body(r) = body(r')$, and any triple pattern $c''\colon (s'',p'',o'') \in body(r)$ is such that $c'' \in \mathcal{C}$, if there exists μ with $body(r)[\mu] \in dChase_k^{\text{safe}}(QS_{\mathcal{C}})$, then $body(r')[\mu] \in dChase_k(QS_{\mathcal{C}})$ (follows from induction hypothesis). Also, since any triple pattern that are in $head(r) \setminus head(r')$ are triple-patterns of the form $c_c\colon (s,p,o)$, such that $c_c \notin \mathcal{C}$. Hence, $c'\colon (s',p',o') \in head(r)[\mu]$ implies that $c'\colon (s',p',o') \in head(r')[\mu]$. Hence, it follows that $S \subseteq dChase_{k+1}(QS_{\mathcal{C}})$. Also, since $dChase_{k+1}(QS_{\mathcal{C}})$ is closed w.r.t. OWL-Horst rules, and since $S \vdash_{\text{owl-horst}} c\colon (\ s,\ p,\ o)$, $c\colon (s,p,o) \in dChase_{k+1}(QS_{\mathcal{C}})$.

The following claim shows that the set of origin context triples are also sound.

Claim (2) If there exists quad $c_c\colon$ (b, originContext, c) $\in dChase^{\text{safe}}(QS_{\mathcal{C}})$, then $c \in originContexts(b)$.

If $c_c\colon$ (b, originContext, c) $\in dChase^{\text{safe}}(QS_{\mathcal{C}})$, there exists $i \in \mathbb{N}$, such that $c_c\colon$ (b, originContext, c) $\in dChase_i^{\text{safe}}(QS_{\mathcal{C}})$, there exists no $j < i$ with $c_c\colon$ (b,originContext, c) $\in dChase_j^{\text{safe}}(QS_{\mathcal{C}})$. But if $c_c\colon$ (b, originContext, c) $\in dChase_i^{\text{safe}}(QS_{\mathcal{C}})$ implies that there exists an $r = body(\vec{x},\vec{z}) \rightarrow head(\vec{x},\vec{f}(\vec{x}))$ $\in aug(R)$, with $c_c\colon$ ($f_i(\vec{x})$, originContext, c) $\in head(\vec{x},\vec{f}(\vec{x}))$, and $\mu \in M$, such that $c_c\colon$ (b, originC--ontext, c) was generated due to application of μ on $aug(r)$, with $b = f_i(\vec{x})[\mu]$. This implies that $c \in cScope(f_i(\vec{x}), head(\vec{x},\vec{f}(\vec{x}))$ (By construction of $aug(r)$). This implies that there exists $c\colon (s,p,o) \in head(\vec{x},\vec{f}(\vec{x}))$, with $s = f_i(\vec{x})$ or $p = f_i(\vec{x})$ or $o = f_i(\vec{x})$. Since according to our assumption i is the first iteration in which $c_c\colon$ (b, originContext, c) is generated, it follows that i is the first iteration in which $c\colon (s,p,o)$ is also generated. This implies that $c \in orginContexts(b)$.

In the following claim we prove the soundness of the descendant triples generated in a safe dChase.

Claim (3) For any two distinct blank nodes b, b' in $dChase^{\text{safe}}(QS_{\mathcal{C}})$, if $c_c\colon$ (b', descendantOf, b) $\in dChase^{\text{safe}}(QS_{\mathcal{C}})$ then b' is a descendant of b.

Since any quad of the form $c_c\colon$ (b', descendantOf, b) $\in dChase^{\text{safe}}(QS_{\mathcal{C}})$ is not an element of $Q_{\mathcal{C}}$, and can only be introduced by an application of a BR $r \in aug(R)$, any quad of the form $c_c\colon$ (b', descendantOf, b) can only be introduced, earliest in the first iteration of $dChase^{\text{safe}}(QS_{\mathcal{C}})$. Suppose $c_c\colon$ (b', descendantOf, b) $\in dChase^{\text{safe}}(QS_{\mathcal{C}})$, then there exists an iteration $i \geq 1$ such that $c_c\colon$ (b', descendantOf, b) $\in dChase_j^{\text{safe}}(QS_{\mathcal{C}})$, for any $j \geq i$. We apply induction on i for the proof.

base case suppose c_c:(b', descendantOf, b) $\in dChase_1(QS_{\mathcal{C}})$ and since $b \neq b'$, then there exists a BR r $\in aug(R)$, $\exists \mu \in M$, such that $body(r)(\vec{x},\vec{z})[\mu]$ $\in dChase_i^{\text{safe}}(QS_{\mathcal{C}})$, and $c_c\colon$ (b', descendantOf, b) $\in head(r)(\vec{x},\vec{f}(\vec{x}))[\mu]$. Then by construction of $aug(r)$, it follows that $b = f_i(\mu(\vec{x}))$, for some $f_i(\vec{x}) \in \vec{f}(\vec{x})$. Also since $b' \neq b$, $b' = \mu(x_i)$, for some $x_i \in \vec{x}$. But since $b = f_i(\mu(\vec{x}))$, which can be rewritten as $b = f_i(\mu(x_1),...,\mu(x_n))$. Hence b' is a descendant of b (by definition).

hypothesis if $c_c\colon$ (b', descendantOf, b) $\in dChase_i(QS_{\mathcal{C}})$, for $1 \leq i \leq k$, then b' is a descendant of b

inductive step suppose $c_c\colon$ (b', descendantOf, b) $\in dChase_{k+1}(QS_{\mathcal{C}})$, then either (i) $c_c\colon$ (b', descendantOf, b) $\in dChase_k(QS_{\mathcal{C}})$ or (ii) $c_c\colon$ (b', descendantOf, b) $\notin dChase_k(QS_{\mathcal{C}})$. Suppose (i) is the case, then by hypothesis, b' is a descendant of b. If (ii) is the case, then either (a) $c_c\colon$ (b', descendantOf, b) is the result of the application of a BR $r \in R$ on $dChase_k^{\text{safe}}(QS_{\mathcal{C}})$ or (b) there exists blank-node $b'' \neq b, b'$, such that $c_c\colon$ (b'', descendantOf, b) is the result of application of a BR $r \in R$ and $c_c\colon$ (b' descendantOf, b'') $\in dChase_k(QS_{\mathcal{C}})$. If (a) is the case, then similar to what we saw in the base case, it follows that b' is a descendant of b, where as if (b) is the case then similar to what we saw in the base case, b'' is a descendant of b, and from hypothesis b' is a descendant of b''. Also since, by definition, relation 'descendant of' is transitive, b' is a descendant of of b.

Suppose the constant **unSafe** $\in dChase^{\text{safe}}(QS_{\mathcal{C}})$, then this implies that there exists an iteration i such that the function unSafeTest on R and $dChase_i^{\text{safe}}(QS_{\mathcal{C}})$ returns True. This implies that, there exists $r = body(r)(\vec{x}, \vec{z}) \to head(r)(\vec{x}, \vec{f}(\vec{x})) \in aug(R)$, $\mu \in M, b, b' \in \mathbf{B}$, $f_i(\vec{x}) \in \vec{f}(\vec{x})$, such that $body(r)(\vec{x}, \vec{z})[\mu] \in dChase_i^{\text{safe}}(QS_{\mathcal{C}})$, $b \in \mu(\vec{x})$, $c_c\colon (b', \text{descendantOf}, b) \in dChase_i^{\text{safe}}(QS_{\mathcal{C}})$ and $\{c \mid c_c\colon (b', \text{originContext}, c) \in dChase_i^{\text{safe}}(QS_{\mathcal{C}})\} = cScope(f_i(\vec{x}), head(r)(\vec{x}, \vec{f}(\vec{x})))$. Then one of the following subcases holds: (a) $b = b'$ and (b) $b \neq b'$.

Suppose if (a) is the case, then as a result of μ being applied to r, leads to the generation of a blank node $b'' = f_i(\mu(\vec{x}))$, such that $originContexts(b'') = cScope(f_i(\vec{x}), head(r)(\vec{x}, \vec{f}(\vec{x}))$. Also since $b \in \mu(\vec{x})$ and by definition of unSafeTest, it follows that $\{c \mid c_c\colon (b, \text{originContext}, c) \in dChase_i^{\text{safe}}(QS_{\mathcal{C}})\} = originContexts(b'')$. By claim 2, it follows that $originContexts(b) = originContexts(b'')$ (♣). Also, trivially b is a descendant of b'' (♡). Also from claim 1, since $body(r)(\vec{x}, \vec{z})[\mu] \in dChase_i^{\text{safe}}(QS_{\mathcal{C}})$, $body(r)(\vec{x}, \vec{z})[\mu]$ is also in $dChase_j(QS_{\mathcal{C}})$ for some iteration j, and hence r is applicable on $dChase_j(QS_{\mathcal{C}})$ for μ, and since applying μ on $head(r)[\vec{x}, \vec{f}(\vec{x})]$, produces a skolem quad in which b'' occurs, and hence $b'' \in \mathbf{B}_{sk}(dChase(QS_{\mathcal{C}}))$. Also since $b \in \mu(\vec{x})$, implies that $b \in \mathbf{B}_{sk}(dChase(QS_{\mathcal{C}}))$. Hence $b, b'' \in \mathbf{B}_{sk}(dChase(QS_{\mathcal{C}}))$(♠). By (♣), (♡), (♠), all the prerequisites of an unsafe quad-system is satisfied, and hence $QS_{\mathcal{C}}$ is unsafe.

Suppose if (b) is the case, then as a result of μ being applied to r, leads to the generation of a blank node $b'' = f_i(\mu(\vec{x}))$, such that $originContexts(b'') = cScope(f_i(\vec{x}), head(r)(\vec{x}, \vec{f}(\vec{x}))$. Also since $b \in \mu(\vec{x})$, and $b'' = f_i(\mu(\vec{x})$, b is a descendant of b''. Also since, by assumptions of unSafeTest, $c_c\colon (b'$, descendantOf, $b) \in dChase_i^{\text{safe}}(QS_{\mathcal{C}})$, by claim 3, it follows that b' is a descendant of b and by transitivity, it follows that b' is a descendant of b'' (♡). Also by assumptions of unSafeTest, it follows that $\{c \mid c_c\colon (b'$, originContext, $c) \in dChase_i^{\text{safe}}(QS_{\mathcal{C}})\} = originContexts(b'')$. By claim 2, it follows that $originContexts(b') = originContexts(b'')$ (♣). Also from claim 1, since $body(r)(\vec{x}, \vec{z})[\mu] \in dChase_i^{\text{safe}}(QS_{\mathcal{C}})$, $body(r)(\vec{x}, \vec{z})[\mu]$ is also in $dChase_j(QS_{\mathcal{C}})$ for some iteration j, and hence r is applicable on $dChase_j(QS_{\mathcal{C}})$ for μ, and since applying μ on $head(r)(\vec{x}, \vec{f}(\vec{x}))$, produces a skolem quad in which b'' occurs, and hence $b'' \in \mathbf{B}_{sk}(dChase(QS_{\mathcal{C}}))$. Also by claim 1, it follows that $b, b' \in \mathbf{B}_{sk}(dChase(QS_{\mathcal{C}}))$. By definition of unsafe quad-systems and by (♣), (♡), (♠), $QS_{\mathcal{C}}$ is unsafe. □

Lemma C.2 (Completeness)**.** *For any quad-system, $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$, if $QS_{\mathcal{C}}$ is unsafe then **unSafe** $\in dChase^{\text{safe}}(QS_{\mathcal{C}})$.*

Proof. We first prove a few supporting claims in order to prove the theorem. The following claim shows that, for safe quad-systems its standard dChase is contained in its safe dChase.

Claim (1) Suppose **unSafe** $\notin dChase^{\text{safe}}(QS_{\mathcal{C}})$, then $dChase(QS_{\mathcal{C}}) \subseteq dChase^{\text{safe}}(QS_{\mathcal{C}})$. We approach the proof by induction on the iterations needed during the dChase computation.

base case since $dChase_0(QS_{\mathcal{C}}) = Q_{\mathcal{C}}$, $dChase_0(QS_{\mathcal{C}}) \subseteq dChase_0^{\text{safe}}(QS_{\mathcal{C}})$.

hypothesis for any $i \leq k$, if **unSafe** $\notin dChase_i(QS_{\mathcal{C}})$, then $dChase_i(QS_{\mathcal{C}}) \subseteq dChase_i^{\text{safe}}(QS_{\mathcal{C}})$.

inductive step since $dChase_{k+1}(QS_{\mathcal{C}}) = owl\text{-}horst\text{-}closure(dChase_k(QS_{\mathcal{C}})) \cup R(dChase_k(QS_{\mathcal{C}})))$, and $dChase_{k+1}^{\text{safe}}(QS_{\mathcal{C}}) = owl\text{-}horst\text{-}closure\,(dChase_k^{\text{safe}}(QS_{\mathcal{C}}) \cup aug(R)(dChase_k^{\text{safe}}(QS_{\mathcal{C}})))$, and since by induction hypothesis, $dChase_k(QS_{\mathcal{C}}) \subseteq dChase_k^{\text{safe}}(QS_{\mathcal{C}})$, it follows that $dChase_{k+1}^{\text{safe}}(QS_{\mathcal{C}}) \supseteq owl\text{-}horst\text{-}closure(dChase_k(QS_{\mathcal{C}}) \cup aug(R)(dChase_k(QS_{\mathcal{C}})))$. Also since by construction, if a BR r is applicable on an $dChase_k(QS_{\mathcal{C}})$, then also $aug(r)$ is applicable as both r and $aug(r)$ have the same head part. Also $aug(r)$ augments more quad-patterns to the head part, application of $aug(r)$ produces at least as many triples as r produces. Hence, we can rewrite the expression derived before as: $dChase_{k+1}^{\text{safe}}(QS_{\mathcal{C}}) \supseteq owl\text{-}horst\text{-}closure(dChase_k(QS_{\mathcal{C}}) \cup R(dChase_k(QS_{\mathcal{C}})))$, which can again be rewritten as: $dChase_{k+1}^{\text{safe}}(QS_{\mathcal{C}}) \supseteq dChase_{k+1}(QS_{\mathcal{C}})$.

Claim (2) For any skolem blank-node b generated in $dChase^{\text{safe}}(QS_{\mathcal{C}})$, and for any $c \in \mathcal{C}$, if $c \in originContexts(b)$, then there exists a quad $c_c\colon (b, \texttt{originContext}, c) \in dChase^{\text{safe}}(QS_{\mathcal{C}})$.

Since, the only way a skolem blank node b gets generated in any iteration i of $dChase^{\text{safe}}(QS_{\mathcal{C}})$, is by the application of a BR $r \in aug(r)$, i.e. when there $\exists r = body(r)(\vec{x},\vec{z}) \rightarrow head(r)(\vec{x},\vec{f}(\vec{x})) \in aug(R), \exists \mu \in M$, such that $body(r)(\vec{x},\vec{z})[\mu] \in dChase^{\text{safe}}_{i-1}(QS_{\mathcal{C}})$, and there exists $f_i(\vec{x}) \in \vec{f}(\vec{x})$, with $b = f_i(\mu(\vec{x}))$. But by construction of $aug(r)$, $head(r)(\ \vec{x}, \vec{f}(\vec{x}))$ also has a quad-pattern of the form c_c: ($f_i(\vec{x})$, `origin-context`, c), for every $c \in cScope(f_i(\vec{x}), dChase^{\text{safe}}_{i-1}(QS_{\mathcal{C}}))$, and hence also a triple of the form c_c: ($f_i(\mu(\vec{x}))$, `origin-context`, c) gets generated, on the application of μ on $head(r)($ $\vec{x}, \vec{f}(\vec{x})$. Since origin context of $b = f_i(\mu(\vec{x}))$ is the set $cScope(f_i(\vec{x}), dChase^{\text{safe}}_{i-1}(QS_{\mathcal{C}}))$, the claim follows.

For the claim below, we introduce the concept of the sub-distance. For any two blank nodes, their sub-distance is inductively defined as:

Definition C.3. *For any two blank nodes b, b', sub-distance(b, b') is defined inductively as:*

- *sub-distance$(b, b') = \infty$, if b is not a descendant of b';*
- *sub-distance$(b, b') = 1$, if $b' = f(t_1, ..., t_n)$ and $b = t_i$;*
- *sub-distance$(b, b') = min\{$sub-distance$(b, t_i)\}+ 1$, if $b' = f(t_1, ..., t_n)$ and $b \neq t_i$ and b is a descendant of b'.*

Claim (3) For any two skolem blank nodes b, b' in $dChase^{\text{safe}}(QS_{\mathcal{C}})$, if b is a descendant of b' then there exists a quad of the form c_c: $(b, \text{descendantOf}, b') \in dChase^{\text{safe}}(QS_{\mathcal{C}})$

Since, from the definition of sub-distance, it can be seen that if b is a descendant of b', then sub-distance$(b, b') \in \mathbb{N}$. We approach the proof by induction on sub-distance(b, b').

base case Suppose sub-distance$(b, b') = 1$, then this implies that $b' = f(t_1, ..., t_n)$ and $b = t_i$. Since by construction, the only way b' is generated, in any iteration i of $dChase^{\text{safe}}(QS_{\mathcal{C}})$, is by the application of a BR, i.e. when there $\exists r = body(r)(\vec{x},\vec{z}) \rightarrow head(r)(\vec{x},\vec{f}(\vec{x})) \in aug(R)$, $\exists \mu \in M$ $body(r)(\vec{x},\vec{z})[\mu] \in dChase^{\text{safe}}_{i}(QS_{\mathcal{C}})$, and there exists an $f_i(\vec{x}) \in \vec{f}(\vec{x})$ with $b' = f_i(\mu(\vec{x}))$ and $b \in \mu(x_i)$, for some $x_i \in \vec{x}$. But by construction of $aug(r)$, $head(r)(\vec{x}, \vec{f}(\vec{x}))$, also has a quad-pattern of the form c_c: (x_i, descendantOf, $f_i(\vec{x})$). Hence application of μ on $head(r)(\vec{x}, \vec{f}(\vec{x}))$, also produces quads of the form c_c: ($\mu(x_i)$, descendantOf, $f_i(\mu(\vec{x}))$), which means that c_c: $(b, \text{descendantOf}, b') \in dChase^{\text{safe}}(\ QS_{\mathcal{C}})$.

hypothesis Suppose sub-distance$(b, b') \leq k$, for some $1 \leq k \in \mathbb{N}$, then c_c: (b, descendantOf, b') $\in dChase^{\text{safe}}(QS_{\mathcal{C}})$.

inductive step Suppose sub-distance$(b, b') = k + 1$, then there exists a $b'' \neq b$, such that $b' = f(t_1, ..., t_n)$, and $t_j = b''$, for some $1 \leq j \leq n$, and b is a descendant of b''. This implies that sub-distance$(b'', b') = 1$, and sub-distance$(b, b'') = k$, and hence by hypothesis c_c: (b, descendantOf, b'') $\in dChase^{\text{safe}}(QS_{\mathcal{C}})$, and c_c: (b'', descendantOf, b') $\in dChase^{\text{safe}}(QS_{\mathcal{C}})$, and since by construction c_c: (descendantOf, `rdf:type`, `owl:TransitiveProperty`) $\in dChase^{\text{safe}}($ $QS_{\mathcal{C}})$. Hence, c_c(b,descendantOf, b') $\in dChase^{\text{safe}}(QS_{\mathcal{C}})$.

Suppose $QS_{\mathcal{C}}$ is unsafe, then by definition, there exists a blank nodes b, b' in $\mathbf{B}_{sk}(\ dChase(QS_{\mathcal{C}}))$, such that b is descendant of b', and $originContexts(b) = originContexts(b')$. By contradiction, if **unSafe** $\notin dChase^{\text{safe}}(QS_{\mathcal{C}})$, then by claim 1, $dChase(QS_{\mathcal{C}}) \subseteq dChase^{\text{safe}}(QS_{\mathcal{C}})$. Since by claim 2, for any $c \in originContexts(b)$, there exists quads of the form c_c: (b, `origin-context`, c) $\in dChase^{\text{safe}}(QS_{\mathcal{C}})$ and for every $c' \in originContexts(b')$, there exists c_c: (b', originContext, c') $\in dChase^{\text{safe}}(\ QS_{\mathcal{C}})$. Since $originContexts(b) = originContexts(\ b')$, it follows that $\{c \mid c_c$: (b, `origin-context`, c) $\in dChase^{\text{safe}}(\ QS_{\mathcal{C}})\} = \{c' \mid c_c$: ($b'$, `origin-context`, c') $\in dChase^{\text{safe}}(QS_{\mathcal{C}})\}$ Also by claim 3, since b is a descendant of b', there exists a quad of the form c_c: (b, descendantOf, b') in $dChase^{\text{safe}}(QS_{\mathcal{C}})$. But by construction of $dChase^{\text{safe}}(QS_{\mathcal{C}})$, there should exist a $b'' \in \mathbf{B}_{sk}(dChase^{\text{safe}}(QS_{\mathcal{C}}))$, $r \in aug(R)$, $\mu \in M$, such that c_c: (b, descendantOf, b'') $\in dChase^{\text{safe}}(QS_{\mathcal{C}})$ and $b'' \in \mu(x_i)$, and $b' = f_i(\mu(\vec{x}))$ and since $\{c \mid c_c$: (b, `origin-context`, c) $\in dChase^{\text{safe}}(QS_{\mathcal{C}})\} = \{c' | c_c$: ($b'$, `origin-context`, c') $\in dChase^{\text{safe}}(QS_{\mathcal{C}})\}$, the method **unSafeTest**($dChase^{\text{safe}}_{i}(QS_{\mathcal{C}}), R)$ should return True, for some i. Hence, it should be the case that **unSafe** $\in dChase^{\text{safe}}($ $QS_{\mathcal{C}})$, which is a contradiction to our assumption. Hence, **unSafe** $\in dChase^{\text{safe}}(QS_{\mathcal{C}})$, if $dChase(QS_{\mathcal{C}})$ is unsafe. □

Theorem 4.11. Follows from lemma C.1 and lemma C.2. □