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

Mathew Joseph a,b,∗

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 4store1, Openlink Virtuoso2, and some of the current popular triple stores like Sesame3, Allegrograph4 internally keep track of the context by storing arrays of four names \((c, s, p, o)\) (further denoted as \(c: (s, p, o)\)), 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.

1The work was done as part of the research project, Contextualized Knowledge Repositories (CKR) for the Semantic Web, funded by FBK-IRST, Trento, Italy.
2http://4store.org
3http://virtuoso.openlinksw.com/rdf-quad-store/
4http://www.openrdf.org/
5http://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, geolocation, and topic. Having defined various contexts in which triples are dispersed, one can declare in a meta-context mc, statements such as mc: (c₁, creator, John), mc: (c₁, expiryTime, “jun-2013”) that talk about the knowledge in context c₁, 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 c₁ contains knowledge about football world cup 2010 and context c₂ 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:

\[ c₁: (x, \text{beat}, \text{Spain}) \land c₂: (x, \text{beat}, \text{Spain}), \]

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 \{c₁: (a, owl:sameAs, b), c₂: (a, owl:disjointFrom, b)\}, and \{c₁: (C, owl:disjointWith, D), c₁: (a, rdf:type, C), c₂: (a, 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 \( \phi, \phi' \) 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 $U$ be the set of URIs, $B$ the set of blank nodes, and $L$ the set of literals. The set $C = U \cup B \cup L$ is called the set of (RDF) constants. Any $(s, p, o) \in C \times C \times C$ is called a generalized RDF triple (from now on, just triple). A graph is defined as a set of triples. A \textit{Quad} is a tuple of the form $c: (s, p, o)$, where $(s, p, o)$ is a triple and $c$ is a URI, called the \textit{context identifier} that denotes the context of the RDF triple. Let $X$ be the set of variables, any element of the set $C^x = X \cup C$ is a \textit{term}. Any $(s, p, o) \in C^x \times C^x \times C^x$ is called a \textit{quad pattern}, and an expression of the form $c: (s, p, o)$, where $(s, p, o)$ is a triple pattern, $c$ a context identifier, is called a \textit{quad pattern}. A \textit{quad-graph} is defined as a set of quads. For any quad-graph $Q$ and any context identifier $c$, we denote by $\text{graph}_Q(c)$ the set $\{(s, p, o) : (s, p, o) \in Q\}$. We denote by $Q_C$ the quad-graph whose set of context identifiers is $C$. For the sake of enabling interoperability between knowledge in different contexts, special rules called Bridge rules have to be provided:

\textbf{Bridge rules (BRs)} Formally, a BR is of the form:

\[ \forall \vec{x} \forall \vec{z} \left[ c_1: t_1(\vec{x}, \vec{z}) \land \ldots \land c_n: t_n(\vec{x}, \vec{z}) \rightarrow \exists \vec{y} \ c_1': t_1'(\vec{x}, \vec{y}) \land \ldots \land c_m': t_m'(\vec{x}, \vec{y}) \right] \quad (1) \]

where $c_1, \ldots, c_n, c_1', \ldots, c_m'$ 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 $\text{body}(r) = \{c_1: t_1(\vec{x}, \vec{z}), \ldots, c_n: t_n(\vec{x}, \vec{z})\}$, and $\text{head}(r) = \{c_1': t_1'(\vec{x}, \vec{y}), \ldots, c_m': t_m'(\vec{x}, \vec{y})\}$.

\textbf{Definition 2.1 (Quad-System).} A quad-system $QS_C$ is defined as a pair $(Q_C, R)$, where $Q_C$ is a quad-graph, whose set of context identifiers is $C$, and $R$ is a set of BRs.

For any quad-graph $Q_C$, its size $|Q_C|$ is the number of quads in $Q_C$, 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 $QS_C = (Q_C, R)$, its size $|QS_C| = |Q_C| + |R|$.

\textbf{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 much hassle, 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 $(IR, IP, IC, IEXT, IS, LV)$, 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, $IS$, 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:

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

For any triple-pattern $(s, p, o)$, and for any function $\sigma$, 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 $\mathcal{I}^C$ and a quad-system $QS_C$ as:

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

1. $\mathcal{I}^c \models_{\text{owl-horst}} \text{graph}_Q(c)$ for each $c \in C$, where $\models_{\text{owl-horst}}$ is the classical satisfaction relation between an OWL-Horst interpretation and a graph.
2. $\mathcal{I}^S^c(a) = \mathcal{I}^S^{c_1}(a)$, if $\mathcal{I}^S^{c_2}(a) \subseteq IR^c$ and $\mathcal{I}^S^{c_2}(a) \subseteq IR^{c_1}$, for any $a \in C$, $c_i, c_j \in C$
3. for each BR $r \in R$ of the form (1) and for each $\sigma \in \Sigma$, if

\[ \mathcal{I}^c \models_{\text{owl-horst}} t_1(\vec{x}, \vec{z})[\sigma], \ldots, \mathcal{I}^c \models_{\text{owl-horst}} t_n(\vec{x}, \vec{z})[\sigma], \]

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

\[ \mathcal{I}^c \models_{\text{owl-horst}} t'_1(\vec{x}, \vec{y})[\sigma'], \ldots, \mathcal{I}^c \models_{\text{owl-horst}} t'_m(\vec{x}, \vec{y})[\sigma'], \]

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

Condition 1 in the above definition ensures that for any model $\mathcal{I}^C$ of a quad-graph, each $\mathcal{I}^c \in \mathcal{I}^C$ is an OWL-Horst model of the set of contexts in $c$. 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 \( \mathcal{T}^c \) such that \( \mathcal{T}^c \models Q_{SC} \) is said to be a model of \( Q_{SC} \). A quad-system \( Q_{SC} \) is said to be consistent if there exists a model \( \mathcal{T}^c \), such that \( \mathcal{T}^c \models Q_{SC} \), and \( Q_{SC} \) is said to be inconsistent if it is not consistent. For any quad-system \( Q_{SC} = \langle Q_C, R \rangle \), it can be the case that \( \text{graph}Q_{SC} \) is OWL-Horst consistent, for each \( c \in C \), whereas \( Q_{SC} \) 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 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 \( Q_{SC} \) entails a quad \( c: (s, p, o) \), in symbols \( Q_{SC} \models c \), iff for any distributed interpretation structure \( \mathcal{T}^c \), if \( \mathcal{T}^c \models Q_{SC} \) then \( \mathcal{T}^c \models \{ \langle c: (s, p, o) \rangle \} \). A quad-system \( Q_{SC} \) entails a quad-graph \( Q'_{Cv} \), in symbols \( Q_{SC} \models Q'_{Cv} \), iff \( Q_{SC} \models c: (s, p, o) \) for any \( c: (s, p, o) \in Q'_{Cv} \). A quad-system \( Q_{SC} \) entails a BR \( r \), iff for any distributed interpretation structure \( \mathcal{T}^c \), if \( \mathcal{T}^c \models Q_{SC} \) then \( \mathcal{T}^c \models \{ \langle \}, \{ r \} \}. \)

For a set of BRs \( R \), \( Q_{SC} \models R \) iff \( Q_{SC} \models r \), for every \( r \in R \). Finally, a quad-system \( Q_{SC} \) entails another quad-system \( Q'_{SC} = \langle Q'_{Cv}, R' \rangle \), in symbols \( Q_{SC} \models Q'_{SC} \) iff \( Q_{SC} \models Q'_{Cv} \) and \( Q_{SC} \models R' \).

### 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 \( \bar{x} \), let \( |\bar{x}| \) denote its size. A CQ, \( Q(\bar{x}) \leftarrow \exists \bar{y}. t_1(\bar{x}, \bar{y}) \land \cdots \land t_n(\bar{x}, \bar{y}) \), where \( t_i \), for \( i = 1, ..., n \), are triple patterns over vectors of variables \( \bar{x} = \langle x_1, ..., x_{|\bar{x}|} \rangle \) and \( \bar{y} = \langle y_1, ..., y_{|\bar{y}|} \rangle \). The variables in \( \bar{x} \) are called free variables, the variables in \( \bar{y} \) are quantified variables. Let \( \bar{a} \) be a vector such that \( a_i \in U \cup L \) and \( |\bar{a}| = |\bar{a}| \); then, \( \bar{x}/\bar{a} \) denote simultaneous substitution of \( x_i \) by \( a_i \), for \( i = 1, ..., |\bar{x}| \). For any query \( Q(\bar{x}) \), \( \bar{x}/\bar{a} \) in \( Q(\bar{x}) \) is denoted by \( Q(\bar{a}) \). Any non-boolean query \( Q(\bar{x}) \) becomes a boolean query after the substitution of \( \bar{x} \) by a tuple of names, \( \bar{a} \), 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 \( Q(\bar{x}) \) is an expression of the form:

\[
Q(\bar{x}) \leftarrow \exists \bar{y}. t_1(\bar{x}, \bar{y}) \land \cdots \land t_n(\bar{x}, \bar{y})
\]

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

For any distributed interpretation structure \( \mathcal{T}^c = \{ I^c \}_{c \in C} \) with \( I^c = \langle \text{IR}^c, \text{IP}^c, \text{IC}^c, \text{IEXT}^c, \text{ICEXT}^c, \text{LV}^c, \text{IS}^c \rangle \), let \( \text{IR}^c = \bigcup_{c \in C} \text{IR}^c \) be called the domain of \( \mathcal{T}^c \). A vector \( \bar{a} \) is an answer for a CCQ \( Q(\bar{x}) \) w.r.t. a distributed interpretation structure \( \mathcal{T}^c = \{ I^c \}_{c \in C} \) in symbols \( \mathcal{T}^c \models Q(\bar{a}) \), iff \( I^c = \text{owl-horst} t_i(\bar{a}, \bar{y})[\mu] \), for \( i = 1, ..., n \), where \( \mu : \{ y_1, ..., y_{|\bar{y}|} \} \rightarrow \text{IR}^c \) is an assignment from set of variables in \( \bar{y} \) to the domain of \( \mathcal{T}^c \). A vector \( \bar{a} \) is a certain answer for a CCQ \( Q(\bar{x}) \) w.r.t. a quad-system \( Q_{SC} \) iff \( \mathcal{T}^c \models Q(\bar{a}) \) for every model \( \mathcal{T}^c \) of \( Q_{SC} \). In this case, we say that \( Q_{SC} \) entails \( Q(\bar{a}) \). Note that the problem of deciding, for any given \( CQ(\bar{x}) \), vector \( \bar{a} \), and a quad-system \( Q_{SC} \), if \( Q_{SC} \models Q(\bar{a}) \) is called the **CCQ entailment problem**, and is the problem primarily studied in this paper. Since \( Q(\bar{a}) \) 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 CCQ \( Q(\bar{a}) \), its chase \( \text{chase}(O) \) has the property:

\[
O \models Q(\bar{a}) \text{ iff } t(\bar{a}, \bar{y})[\mu] \in \text{chase}(O), \text{ for all } t(\bar{a}, \bar{y}) \in Q(\bar{a}), \text{ where } \mu : \bar{y} \rightarrow C.
\]

We extend the notion of chase to a quad-system, which we call a distributed chase, abbreviated as \( d\text{Chase} \). In the following, we show how the \( d\text{Chase} \) of a quad-system can be constructed.

#### 3.1. \( d\text{Chase} \) of a Quad-System

For any BR \( r \), we apply skolemization that replaces \( \bar{y} \) in \( r \) with \( \bar{f}^r \), where \( \bar{f}^r = \langle f_1^r, ..., f_{|\bar{y}|}^r \rangle \), is vector of globally unique Skolem functions such that each \( f_i^r : C[|\bar{x}|] \rightarrow B_i^r \), \( B_i^r \subseteq B \) is a fresh set of blank nodes. Intuitively, \( f_i^r(\bar{x}) \) gives a fresh blank node for every distinct input vector \( \bar{a} \). 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(x, z), ..., c_n : t_n(x, z) \rightarrow
c'_1 : t'_1(x, \tilde{f}(\tilde{x})), ..., c'_m : t'_m(\tilde{x}, \tilde{f}(\tilde{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(x, z), ..., c_n : t_n(x, z) \rightarrow c'_1 : t'_1(x, \tilde{f}(\tilde{x}))

... 

c_1 : t_1(x, z), ..., c_n : t_n(x, z) \rightarrow c'_m : t'_m(\tilde{x}, \tilde{f}(\tilde{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_C and BR r of the form (3), application of r on Q_C, denoted by \( r(Q_C) \), is given as:

\[
r(Q_C) = \bigcup_{\mu \in M} \left\{ c'_1 : t'_1(\mu(\tilde{x}), \tilde{f}(\mu(\tilde{x}))), ..., c'_m : t'_m(\mu(\tilde{x}), \tilde{f}(\mu(\tilde{x}))), c_1 : t_1(\mu(\tilde{x}), \mu(\tilde{z})), ..., c_n : t_n(\mu(\tilde{x}), \mu(\tilde{z})) \in Q_C \right\}
\]

For any set of rules R, application of R on Q_C is given as:

\[
R(Q_C) = \bigcup_{r \in R} r(Q_C),
\]

for any quad-graph Q_C, we define:

\[
\text{owl-horst-closure}(Q_C) = \bigcup_{c \in C} \{ c : (s, p, o) \mid (s, p, o) \in \text{owl-horst-closure}(\text{graph}_C(c)) \}
\]

For any quad-system Q_SC = (Q_C, R), 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 \( d\text{Chase}_0(Q_SC) = \text{owl-horst-closure}(Q_C) \), \( d\text{Chase}_{i+1}(Q_SC) = \text{owl-horst-closure}(d\text{Chase}_i(Q_SC) \cup R_I(d\text{Chase}_i(Q_SC))) \), if \( R_I(d\text{Chase}_i(Q_SC)) \not\subseteq d\text{Chase}_i(Q_SC) \);

\[
d\text{Chase}_{i+1}(Q_SC) = \text{owl-horst-closure}(d\text{Chase}_i(Q_SC) \cup R_F(d\text{Chase}_i(Q_SC))) \], otherwise.

\( d\text{Chase} \) of Q_SC, denoted by \( d\text{Chase}(Q_SC) \), is given as:

\[
d\text{Chase}(Q_SC) = \bigcup_{i \in \mathbb{N}} d\text{Chase}_i(Q_SC)
\]

It can be noted that, if there exists \( i \) such that \( d\text{Chase}_i(Q_SC) = d\text{Chase}_{i+1}(Q_SC) \), then, \( d\text{Chase}(Q_SC) = d\text{Chase}_i(Q_SC) \). Any iteration \( i \), such that \( d\text{Chase}_i(Q_SC) \) is computed by the application of the set of (non-)generating BRs, \( R_F \) (resp. \( R_I \)), on \( d\text{Chase}_{i-1}(Q_SC) \) is called a generating iteration (resp. non-generating iteration).

In general, for any quad-system Q_SC = (Q_C, R), its \( d\text{Chase} \) need not be unique, since final constructed \( d\text{Chase} \) 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_C and rules in R. In each \( d\text{Chase} \) iteration, applying BRs respecting this order, and also for each BR \( r \), applying assignments to \( r \) in this order, one can construct a unique \( d\text{Chase} \) for any quad-system.

From now on, we assume that for any quad-system Q_SC, d\text{Chase}(Q_SC) denotes its unique d\text{Chase} constructed using the above mentioned procedure. We call the sequence \( d\text{Chase}_0(Q_SC) \), \( d\text{Chase}_1(Q_SC) \), ..., the \( d\text{Chase} \) sequence of Q_SC. The following lemma shows that, for any quad-system the result of a single generating iteration and any subsequent non-generating iterations in its \( d\text{Chase} \) sequence, causes only a worst case exponential blow up in size.

**Lemma 3.1.** For a quad-system Q_SC = (Q_C, R), then the following holds: (i) if \( i \in \mathbb{N} \) is a generating iteration, then \( |d\text{Chase}_i(Q_SC)| = O(|d\text{Chase}_{i-1}(Q_SC)|^{1|R|}) \), (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 \( |d\text{Chase}_{i+j}(Q_SC)| = O(|d\text{Chase}_{i-1}(Q_SC)|^{1|R|}) \), (iii) for any iteration \( k \), \( d\text{Chase}_k(Q_SC) \) can be computed in time \( O(|d\text{Chase}_{k-1}(Q_SC)|^{1|R|}) \).

**Proof.** (sketch)

(i) \( R \) can be applied on \( d\text{Chase}_{i-1}(Q_SC) \) by grounding \( R \) to the set of constants in \( d\text{Chase}_{i-1}(Q_SC) \), the number of such groundings is of the order \( O(|d\text{Chase}_{i-1}(Q_SC)|^{1|R|}) \), \( |R|d\text{Chase}_{i-1}(Q_SC) \rangle = O(|R| * |d\text{Chase}_{i-1}(Q_SC)|^{1|R|}) \). Since OWL-horst closure only increases the size polynomially [24], \( |d\text{Chase}_i(Q_SC)| = O(|d\text{Chase}_{i-1}(Q_SC)|^{1|R|}) \).
ii) From (i) we know that \( |R(d\text{Chase}_{i-1}(QS_c))| = O(|d\text{Chase}_{i-1}(QS_c)|^{4|R|}) \). 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 \( O(4 \, |d\text{Chase}_{i-1}(QS_c)|^{R}) \). Since only these many constants can appear in positions \( c, s, p, o \) of any quad generated in the subsequent iterations, the size of \( d\text{Chase}_{i+1}(QS_c) \) can only increase polynomially, which means that \( |d\text{Chase}_{i+1}(QS_c)| = O(|d\text{Chase}_{i-1}(QS_c)|^{|R|}) \).

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

In the following, we give a few computational characteristics of quad-systems whose BRs are of the form (1), which we call un\( restricted \) 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 problems arise 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 \( v \to \bar{w} \), where \( v \in V \), \( \bar{w} \) 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 \( r \to \bar{v} \in P_1 \cup P_2 \), with \( \bar{v} = w_1w_2w_3..w_n \), is encoded as a BR of the form:

\[
c: (x_1, w_1, x_2), c: (x_2, w_2, x_3), ..., c: (x_n, w_n, x_{n+1})
\]

\[
\rightarrow c: (x_1, v, x_{n+1})
\]

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

\[
c: (x, \text{rdf:type}, C) \rightarrow \exists y: c: (x, t_i, y),
\]

\[
c: (y, \text{rdf:type}, C)
\]

and \( Q_c \) contains only the triple:

\[
c: (a, \text{rdf:type}, C)
\]

It can be observed that:

\[
QS_c \models \exists y: c: (a, S_1, y) \wedge c: (a, S_2, y) \iff \quad L(G_1) \cap L(G_2) \neq \emptyset
\]

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

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

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

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

**Definition 4.1 (genExp).** For any constant $k \in d\text{Chase}(QS_C)$, its generating expression $\text{genExp}(k)$ is defined inductively as:

- $\text{genExp}(k) = k$, for any $k \in \mathbf{C}(QS_C)$,
- $\text{genExp}(k) = f(\text{genExp}(k_1), \ldots, \text{genExp}(k_n))$, if $k \in B_{sk}(d\text{Chase}(QS_C))$, generated by a Skolem function $f$ using the vector of constants $(k_1, \ldots, k_n)$.

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

**Example 4.2.** Consider the quad-system $(Q_C, R)$, where $Q_C = \{c_1 : (a, b, c)\}$, and suppose the skolemization $sk(R)$ of $R$ is the following set:

$$sk(R) = \left\{ c_1 : (x_1, x_2, x_3), \rightarrow c_2 : (x_1, x_2, f_1(x_1, x_2)), \rightarrow c_3 : (x_1, x_2, f_1(x_1, x_2)) \right\}$$

**Example 4.3 (Origin-contexts).** For any quad-system $QS_C$, and for any Skolem blank node $b \in B_{sk}(d\text{Chase}(QS_C))$, its origin-contexts, is given as originContexts ($b$) = $\{ c \mid \exists i. c : (s, p, o) \in d\text{Chase}_i(QS_C), s = b \lor p = b \lor o = b, \exists j < i, \exists i'. c : (s', p', o') \in d\text{Chase}_{i'}(QS_C), s' = b \lor p' = b \lor o' = b \}$.

Intuitively, origin-contexts for a Skolem blank node $b$ is the set of contexts in which triples containing $b$ are first generated, during $d\text{Chase}$ construction. Note that there can be multiple contexts to which $b$ can simultaneously be generated. For the quad-system and $d\text{Chase}$, presented in example 4.2,
originContexts(_b1) = \{c_2, c_3\}, originContexts(_b2) = \{c_3\}. Note that in Fig. 1, the origin-contexts of _b1 and _b2 are shown along with their node labels.

**Definition 4.4** (Unsafe quad-systems). A quad-system \(QS_C\) is said to be unsafe, if its dChase dChase(\(QS_C\)), contains blank nodes \(b \neq b'\), with \(b, b' \in B_{sk}(dChase(QS_C))\), 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 \(QS_C = \{Q_C, \Gamma\}\), whose dChase is dChase(\(QS_C\)), any \(b \in B_{sk}(dChase(QS_C))\) 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 \(QS_C\), and for any blank node \(b \in B_{sk}(dChase(QS_C))\), its descendance graph is a directed acyclic graph (DAG).

**Proof.** By construction, as there exists no descendant for any constant \(k \in C(QS_C)\), there cannot be any out-going edge from any such \(k\). Hence, any member of \(C(QS_C)\) cannot be involved in cycles. Hence, the only members that can be involved can be the members of \(C(dChase(QS_C)) - C(QS_C) = B_{sk}(dChase(QS_C))\). But if there exists a \(b \in B_{sk}(dChase(QS_C))\), 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 \(QS_C\) is unsafe, which is a contradiction. 

Since the descendance graph \(G\) of any Skolem blank node \(b \in B_{sk}(dChase(QS_C))\) 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 descendant 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-degree of \(v\).
Proof. The property above is exploited to show that there exist cycles in graph, and at some point reaches a node with no children. For instance the unrolling of the dependence 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 $QS_C$, and any Skolem blank node in $dChase(QS_C)$, the unrolling (Algorithm 1) of its dependence graph results in a tree $t$ such that:

1. any leaf node of $t$ is from the set $C(QS_C)$,
2. any non-leaf node of $t$ is from the set $B_{sk}(dChase(QS_C))$,
3. order($t$) $\leq$ max{|ar($f_i$)|fi 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. Since any node $n$ in the dependency graph is such that $n \in C(dChase(QS_C))$, and since $C(dChase(QS_C)) = C(QS_C) \cup B_{sk}(dChase(QS_C))$. Since any member $m \in C(QS_C)$ cannot have descendants and since any non-leaf node has children, $m$ cannot be a non-leaf node. Hence, non-leaf nodes should be from $B_{sk}(dChase(QS_C))$.

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 a Skolem function symbol occurring in $sk(R)$, which implies that outdegree($n$) $=$ ar($f$).

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.

Lemma 4.7. For any safe quad-system $QS_C = \{Q_C, R\}$, the following holds: (i) the $dChase$ size $|dChase(QS_C)| = O(2^{2^{2|QS_C|}})$, (ii) $dChase(QS_C)$ can be computed in 3EXPTIME, (iii) if $|R|$ is fixed to a constant, then $dChase(QS_C)$ is a polynomial in $|QS_C|$ 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 $C$, since there cannot be paths in $t$ that contain nodes with same $C \subseteq C$ 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 $O(m^{2^{|R|}})$ leaf nodes and $O(m^{2^{|R|}})$ inner nodes, and since each of the children can be elements in $C(QS_C)$, the number of such trees are clearly polynomial in $C(QS_C)$, hence bounds the number of Skolem blank nodes generated in $dChase$ construction.

(ii) From (i) $|dChase(QS_C)|$ is triply exponential in $|QS_C|$, and since each iteration add at-least one quad to its $dChase$, the number of iterations are bounded polynomially in $|QS_C|$. Also, by lemma 3.1 any iteration $i$ can be done in time $O(|dChase_{i-1}(QS_C)|^{|R|})$. Since using (i) $|dChase_{i-1}(QS_C)| = O(2^{2^{2|QS_C|}})$, each iteration $i$ can be done in time $O(2^{|R|2^{2|QS_C|}})$. Also, as number of iterations is triple exponential, computing $dChase(QS_C)$ 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 any constant in $C(QS_C)$. Hence, the number of skolem blank nodes generated is $O(|C(QS_C)|^z)$. Hence, the set of constants in $dChase(QS_C)$ is a polynomial in $|QS_C|$, and also is $|dChase(QS_C)|$.

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

(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 \( QS_C \). Since \(|CQ()|\) is fixed to a constant, and from (†), binding the set of constants in \( dChase(QS_C) \) on \( CQ() \) still gives a number of bindings that is worst case polynomial in the size of \(|QS_C|\). 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 \( |dChase(QS_C)| = O(2^{2^{|QS_C|}}) \) (†), from (†) and (‡), binding the set of constants in \( dChase(QS_C) \) to \( CQ() \) amounts to \( O(2^{|CQ()|2^{|QS_C|}}) \) bindings. Since the dChase is triple exponential in \(|QS_C|\), 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):

\[ \text{ASPACE}(f(n)) = \text{DTIME}(2^{O(f(n))}) \]

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 \( M = (Q, \Sigma, \Delta, q_0) \), where

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

A (universal/existential) configuration is a word \( \alpha \in \Sigma^*Q^*U^*\Sigma^*/\Sigma^*E^*\Sigma^* \). A configuration \( \alpha_1 \) is a successor of the configuration \( \alpha_1 \), if one of the following holds:

1. \( \alpha_1 = \bar{w}_1 q \sigma \sigma \bar{w}_r \) and \( \alpha_2 = \bar{w}_1 q' \sigma \sigma \bar{w}_r \), if \( (q, \sigma, q', R) \in \Delta \), or
2. \( \alpha_1 = \bar{w}_1 q \sigma \sigma \bar{w}_r \) and \( \alpha_2 = \bar{w}_1 q' \sigma \sigma \bar{w}_r \), if \( (q, \sigma, q', \sigma, R) \in \Delta \), or
3. \( \alpha_1 = \bar{w}_1 q \sigma \sigma \bar{w}_r \) and \( \alpha_2 = \bar{w}_1 q \sigma \sigma \bar{w}_r \), if \( (q, \sigma, q', \sigma, \sigma, \sigma, \Delta) \in \Delta \).

where \( q, q' \in Q, \sigma, \sigma', \sigma_1, \sigma_2 \in \Sigma \), and \( \bar{w}_1, \bar{w}_r \in \Sigma^* \). Suppose we put bound on the on the number of tape cells of an ATM by a value \( n \), since each configuration \( \hat{c} \) can be represented in size \(|\hat{c}| = n + 1\), the number of possible configurations is bounded by \( O(2^{n+1}) \). A configuration \( \hat{c} = \bar{w}_1 q \bar{w}_r \) is an accepting configuration iff

- \( q \in U \), and all successor configurations of \( \hat{c} \) 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 $O(2^{2^n})$.

Simulating ATMs using Safe Quad-Systems Consider an ATM $M = \langle Q = U \cup E, \Sigma, \delta, q_0 \rangle$, and a string $w$, with $|w| = n$. Since the number of storage cells is doubly exponentially bounded, we first construct a quad-system $Q^M_C = (Q^M, R)$, where $C = \{c_0, c_1, \ldots, c_n\}$, note that $|C| = |\vec{w}| + 1$. 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 $Q^M_C$ be initialized with the following quads:

$$
c_0: (k_0, \text{rdf:type}, R), c_0: (k_1, \text{rdf:type}, R),
$$

$$
c_0: (k_0, \text{rdf:type}, \text{min}_0), c_0: (k_1, \text{rdf:type}, \text{max}_0), c_0: (k_0, \text{succ}_0, k_1)
$$

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:

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

The combination of minimal element with the minimal element (elements of type $\text{min}_i$) 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 $\text{max}_i$) in $c_i$ create the maximal element of $c_{i+1}$

$$
c_{i+1}: (x_0, x_0, x_1), c_i: (x_0, \text{rdf:type}, \text{min}_i) \rightarrow c_{i+1}: (x_1, \text{rdf:type}, \text{min}_i+1)
$$

$$
c_{i+1}: (x_0, x_0, x_1), c_i: (x_0, \text{rdf:type}, \text{max}_i) \rightarrow c_{i+1}: (x_1, \text{rdf:type}, \text{max}_i+1)
$$

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

$$
c_1: (x_1, \text{succ}_i, x_2), c_{i+1}: (x_0, x_1, x_3),
$$

$$
c_{i+1}: (x_0, x_2, x_4) \rightarrow c_{i+1}: (x_3, \text{succ}_{i+1}, x_4)
$$

$$
c_{i+1}: (x_1, \text{succ}_i, x_2), c_{i+1}: (x_1, x_3, x_5), c_{i+1}: (x_2, x_4, x_6)
$$

$$
c_i: (x_3, \text{rdf:type}, \text{max}_i), c_i: (x_4, \text{rdf:type}, \text{min}_i) \rightarrow c_{i+1}: (x_5, \text{succ}_{i+1}, x_6)
$$

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

$$
c_n: (x_0, \text{succ}_n, x_1) \rightarrow c_n: (x_0, \text{succ}, x_1)
$$

$$
c_n: (x_0, \text{succ}, x_1) \rightarrow c_n: (x_0, \text{succ}_x, x_1)
$$

$$
c_n: (x_0, \text{succ}_x, x_1) \rightarrow c_n: (x_1, \text{succ}, x_2)
$$

$$
\rightarrow c_n: (x_0, \text{succ}, x_2)
$$

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

$$(x_0, \text{head}, x_1)$$

denotes the fact that in configuration $x_0$, the head of the ATM is at cell $x_1$.

$$(x_0, \text{state}, x_1)$$

denotes the fact that in configuration $x_0$, the ATM is in state $x_1$.

$$(x_0, \sigma, x_1)$$

for each $\sigma \in \Sigma$, which denote the fact that in configuration $x_0$, the cell $x_1$ contains $\sigma$.

$$(x_0, \text{succ}, x_1)$$

denotes the linear order between cells of the tape.

$$(x_0, \text{succ}, x_1)$$

denotes the transitive closure of $\text{succ}$. 

$$(x_0, \text{conSucc}_3, x_1)$$

to denote the fact that $x_1$ is a successor configuration of $x_0$ by the transition $\delta$.

$$(x_0, \text{rdf: type}, \text{Accept})$$

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 alphabets $\sigma \neq \sigma'$ are on the same cell, we encode this using the following set of axioms

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

Note that $\text{owl:disjointWith}$ relation is present even in weak logics such as OWL-Horst.
The rules above are instantiated for every $\sigma$ for every $\delta$.

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

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

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

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

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

$c_n: (x_0, head, x_i), c_n: (x_0, conSucc_\delta, x_1), c_n: (x_j, succ, x_i), c_n: (x_0, \sigma, x_j) \rightarrow c_n: (x_1, \sigma, x_j)$

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 $q_e \in E$,

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

For each universal state $q_u \in U$, 

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

Finally since $M$ accepts $\bar{w}$ iff only if the initial configuration $I_0 = q_0 \bar{w} \square$ is an accepting configuration. Hence, $I_0$ is accepting iff $QS^M \models c: (I_0, rdf:type, Accept)$. 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 $dChase_i(QS_C) = dChase_{i+1}(QS_C)$, the dChase computation stops with a completed dChase. Since all the additional quads produced for accounting information, uses a distinguished context identifier $c_c \notin C$, 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: $cScope(t, Q) = \{c | c: (s, p, o) \in Q, s = t \lor p = t \lor o = t\}$. 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 $QS_C = (Q_C, R)$, let $sk(R)$ be the skolemization of $R$ and $c_e$ be an arbitrary context identifier such that $c_e \notin C$, then for $r \in sk(R)$, we define transformation $aug(r)$ as:

$$aug(r) = \bigwedge_{q \in body(r)(\vec{x}, \vec{z})} q \rightarrow \bigwedge_{q' \in head(r)(\vec{x}, \vec{f}(\vec{x}))} q' \land \forall c_e : (x_i, descendantOf, f_i(\vec{x})) \land c_e : (f_i(\vec{x}), descendantOf, f_i(\vec{x})) \land c_e : (f_i(\vec{x}), originContext, c_i)$$

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. $c_e : (x_i, descendantOf, f_i(\vec{x}))$, 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 $f_i(\vec{x})$, 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 $c_e : (a_i, descendantOf, f_i(\vec{a}))$ are also produced, and in this way, keeps track of the descendants of any Skolem blank node produced.

2. $c_e : (f_i(\vec{x}), descendantOf, f_i(\vec{x}))$, in order to maintain also the reflexivity of ‘descendantOf’ relation.

3. $c_e : (f_i(\vec{x}), originContext, c_i)$, 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 containing $f_i(\vec{x})$, 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 $c_e : (f_i(\vec{a}), originContext, c_i)$ 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 $\exists r = body(r)(\vec{x}, \vec{z}) \rightarrow head(r)(\vec{x}, \vec{f}(\vec{x})) \in R, \exists \mu \in M, \exists b, b' \in B, \exists f_i(\vec{x}) \in \vec{f}(\vec{x})$ with the following being satisfied:

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

Intuitively, $unsafeTest$ 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 $\mu$ with Skolem blank node $b \in \mu(\vec{x})$, such that $r$ is applicable on $Q$ using $\mu$, and when $\mu$ 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 $Q_C$ is defined as:

$$R^{safe}(Q_C) = \begin{cases} unsafe, & \text{If } unsafeTest(R, Q_C) = True \\ \bigcup_{r \in aug(R)} r(Q_C), & \text{Otherwise} \end{cases}$$

where $unsafe$ is a constant that is generated, if in any iteration, the safety condition is violated. For any quad-system $QS_C = (Q_C, R)$, we define its $safe$ dChase $dChase^{safe}(Q_C)$ as follows:

$$dChase^{safe}_0(Q_C) = owl-horst-closure(Q_C \cup c_e : (descendantOf, rdf:type, owl:TransitiveProperty))$$

$$dChase^{safe}_{i+1}(Q_C) = owl-horst-closure(dChase^{safe}_i(Q_C) \cup R^{safe}(dChase^{safe}_i(Q_C)))$$

$$dChase^{safe}(Q_C) = \bigcup_{i \in \mathbb{N}} dChase^{safe}_i(Q_C)$$

If there exists $i$ such that $dChase^{safe}_i(Q_C) = dChase^{safe}_{i+1}(Q_C)$, then,

$$dChase^{safe}(Q_C) = dChase^{safe}_i(Q_C).$$

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 $QS_C = (Q_C, R)$, the constant $unsafe \in dChase^{safe}(Q_C)$, iff $QS_C$ is unsafe.
Lemma 5.1. For any Horn quad-system $Q_S^c = \langle Q_c, R \rangle$, the following holds: (i) $|\text{chase}(Q_S^c)| = O(|Q_S^c|^4)$ (ii) $\text{chase}(Q_S^c)$ can be computed in EXPTIME (iii) If $|R|$ is fixed to be a constant, $\text{chase}(Q_S^c)$ can be computed in PTIME.

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

(ii) Since from (i) $|\text{chase}(Q_S^c)| = O(|Q_S^c|^4)$, and in each iteration of the chase at least one new quad should be added, the number of iterations cannot exceed $O(|Q_S^c|^4)$. Since by lemma 3.1, each iteration $i$ of chase computation requires $O(|\text{chase}_{i-1}(Q_S^c)|^{|R|^i})$ time, and $|R| \leq |Q_S^c|$, time required for each iteration is of the order $O(2^{|Q_S^c|})$ 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 $O(|\text{chase}_{i-1}(Q_S^c)|^{|R|^i})$. Since $|R|$ is fixed to a constant, application of $R$ can be done in PTIME. Also we know that owl-horst-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 $|Q_S^c|$, computing chase is in PTIME. 

\[ \square \]

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 $G = (V, E)$, 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 $G = (V, E)$, a source node $s$ and a target node $t$. We create a quad-system $Q_S^c = \langle Q_c, \emptyset \rangle$, where instance set (corresponds to A-box or Data) of $Q_c$, $\text{Abox}(Q_c) = \{c: (v, edge, v') | (v, v') \in E \} \cup \{c: (s, \text{rdf:type}, A)\}$, the constant sized schema set (corresponds to T-box) of $Q_c$, i.e. $\text{Tbox}(Q_c) = \{A \subseteq \text{edge}, A\}$, 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 \( QS_c \models c: (t, \text{rdf:} \text{type}, A) \) 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 \( QS_c \), its chase \( \text{chase}(QS_c) \), can be computed in EXPTIME. Also by lemma 5.1, its chase size \( |\text{chase}(QS_c)| \) is a polynomial w.r.t \( |QS_c| \). 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 \( \text{chase}(QS_c) \). The number of such groundings can at most be \( |\text{chase}(QS_c)||CQ()| \).

Since \( |\text{chase}(QS_c)| \) is polynomial in \( QS_c \), there are an exponential number of groundings w.r.t \( |CQ()| \). Since containment of each of these groundings can be checked in \( \text{chase}(QS_c) \) in PTIME, as \( |\text{chase}(QS_c)| \) is a polynomial w.r.t \( |QS_c| \), 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.

**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 \( QS_c \models CQ() \) be called DP.

(Membership) For any \( QS_c \) whose rules are of restricted Horn-type, by lemma 5.1, its \( \text{chase}(QS_c) \) can be computed in PTIME in the size of \( QS_c \) and \( \text{chase}(QS_c) \) has a polynomial number of constants. Hence, if we guess an assignment \( \mu \) for all the existential variables in CCQ \( CQ() \), to the set of constants in \( \text{chase}(QS_c) \). Then, one can evaluate the CCQ by checking if \( c: (s, p, o) \in \text{chase}(QS_c) \), for each \( c: (s, p, o) \in CQ()|\mu| \), which can be done in time \( O(|CQ| \times |\text{chase}(QS_c)|) \), 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 \( G = (V, E) \), where \( V = \{v_1, \ldots, v_n\} \) 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 \( \text{graph}_{Q_c}(c) \) 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, \ldots, v_n \wedge (v, v') \in E \]

\[ \{ c: (v, edge, v') \wedge c: (v', edge, v) \} \]. 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] Bouquet 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 \( t: (s, p, o) \), 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: \( a: (s, p, o) \), 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 Data-log 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 sub-
languages 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 De-
scription Logic Rules [37], where the authors deal with
rules that can be totally internalized by a DL knowl-
edge 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 answer-
ing over contextualized RDF knowledge. We show
that the problem in general is undecidable, and present
few decidable classes of quad-systems. Table 1 dis-
plays 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 conjuc-
tion and existential quantifiers with decidability guar-
antees. We view the semantics and the results obtained
in this paper as a general foundation for contextual rea-
soning and query answering over contextualized RDF
knowledge formats such as Quads, and can straight-
forwardly 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, Uni-
versity of Trento), Loris Bozzato (FBK-IRST), and
Francesco Corcoglioniti (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

provenance and trust,” in Proc. of the 14th int’l. conf. on WWW,
[3] B. Schauelcr, S. Sizov, S. Staab, and D. T. Tran, “Querying for
meta knowledge,” in WWW ’08: Proceeding of the 17th interna-
tional conference on World Wide Web, (New York, NY, USA),
simulating Information from Peer Sources,” J. Data Semantics,
"Datalog+t+/: A Family of Logical Knowledge Representation
and Query Languages for New Applications," in Logic in Com-
puter Science (LICS), 2010 25th Annual IEEE Symposium on,
pp. 228 –242, july 2010.
[8] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and
H. Stickenschmidt. C-OWL: Contextualizing Ontologies. In
[9] F. Giunchiglia and C. Ghidini. Local models semantics, or con-
textual reasoning = locality + compatibility. Artificial Intelli-
RDF knowledge. In Proc. of Workshop on Modular Ontologies
(WOMO-2011), 2011.
tories for the semantic web. Web Semantics: Science, Services
Dependencies.” In JCALP, pages 73–85, 1981.
the Complexity Lines for Generalized Guarded Existential
Frontiers Of Combining Systems 2, Studies in Logic and Com-
mediation in peer data management systems,” in ICDE,
junctive queries under functional and inclusion dependencies,”
Addison-Wesley, 1995.
Connections of Abstract Description Systems,” Artificial Intelli-
[20] S. Klarman and V. Gutiérrez-Basulto, “Two-dimensional de-
scription logics for context-based semantic interoperability,” in

<table>
<thead>
<tr>
<th>Quad-System Fragment</th>
<th>Chase size w.r.t input quad-system</th>
<th>Data Complexity of CCQ entailment</th>
<th>Combined Complexity of CCQ entailment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unrestricted Quad-Systems</td>
<td>Infinite</td>
<td>Undecidable</td>
<td>Undecidable</td>
</tr>
<tr>
<td>Safe Quad-Systems</td>
<td>Triple exponential</td>
<td>PTIME-complete</td>
<td>3EXPTIME-complete</td>
</tr>
<tr>
<td>Horn Quad-Systems</td>
<td>Polynomial</td>
<td>PTIME-complete</td>
<td>EXPTIME-complete</td>
</tr>
<tr>
<td>Restricted Horn Quad-Systems</td>
<td>Polynomial</td>
<td>PTIME-complete</td>
<td>NP-complete</td>
</tr>
</tbody>
</table>

Table 1
Complexity info for various quad-system fragments
as OWL restrictions (universal, existential, value claims), 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 about using the OWL-Horst semantics. An OWL-Horst interpretation structure is a tuple \( \langle \text{IR}, \text{IP}, \text{IC}, \text{IEXT}, \text{IS}, \text{LV} \rangle \), where \( \text{IR} \) is the object domain, \( \text{IP} \subseteq \text{IR} \) is the property domain, \( \text{IC} \subseteq \text{IR} \) is the class domain, \( \text{IEXT} \), the property extension function, \( \text{IS} \), the term interpretation function, and \( \text{LV} \subseteq \text{IR} \), 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, \( \text{owl-horst-closure}(g) \), 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
\[
\text{can}_{\text{owl-horst}}(g) = \langle \text{IR}^g, \text{IP}^g, \text{IC}^g, \text{IEXT}^g, \text{ICEXT}^g, \text{IS}^g, \text{LV}^g \rangle
\]
is an OWL-Horst interpretation structure, constructed as follows:

- \( \text{LV}^g = \{ l | l \text{ is a plain literal and } l \text{ occurs in } \text{owl-horst-closure}(g) \} \cup \{ dv(l) | l \text{ is a datatyped literal occurring in } \text{owl-horst-closure}(g) \}, \text{where } \text{dv}(l) \text{ is the data value of } l \}
- \( \text{IP}^g = \{ P | (P, \text{rdf:type}, \text{rdf:Property}) \in \text{owl-horst-closure}(g) \} \)
- \( \text{IC}^g = \{ C | (C, \text{rdf:type}, \text{rdfs:Class}) \in \text{owl-horst-closure}(g) \} \)
- \( \text{IR}^g = \{ l | l \text{ occurs in } \text{owl-horst-closure}(g) \} \cup \text{IP}^g \cup \text{IC}^g \cup \{ a | (a, \text{rdf:type}, \text{rdfs:Resource}) \in \text{owl-horst-closure}(g) \} \)
- \( \text{IS}^g = \{ \text{IR}^g \cup \text{IP}^g \cup \text{IC}^g \cup \{ a | (a, \text{rdf:type}, \text{rdfs:Resource}) \in \text{owl-horst-closure}(g) \} \} \cup \{ (l, \text{dv}(l)) | l \text{ is a datatyped literal occurring in } \text{owl-horst-closure}(g) \}, \text{where } \text{dv}(l) \text{ is the data value of } l \}

- for every \( P \in \text{IP}^g \), \( \text{IEXT}^g(P) = \{ (s, o) | (s, P, o) \in \text{owl-horst-closure}(g) \} \)
- for every \( C \in \text{IC}^g \), \( \text{ICEXT}^g(C) = \{ (a, \text{rdf:type}, C) | (a, \text{rdf:type}, C) \in \text{owl-horst-closure}(g) \} \)

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 \( (a, \text{owl:sameAs}, b) \) and \( (a, \text{owl:differentFrom}, b) \). Any graph, \( g \), is said to be OWL-Horst inconsistent, if \( g \not\models_{\text{owl-horst}} \text{false} \), and otherwise said to be OWL-Horst consistent, where \( g \models_{\text{owl-horst}} \) is the OWL-Horst entailment relation. We denote by \( g \|_{\text{owl-horst}} h \) iff \( g \models_{\text{owl-horst}} h \).

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 \( \text{chase}_{e-1}(Q_S) \) by grounding variables in \( r \) to the set of constants in \( \text{chase}_{e-1}(Q_S) \), the number of such groupings is of the order \( O(|\text{chase}_{e-1}(Q_S)|^4) \). Hence, \( |r| |\text{chase}_{e-1}(Q_S)| = O(l^{4} |\text{chase}_{e-1}(Q_S)|^4) \) and \( |R| = O(l^{4} |\text{chase}_{e-1}(Q_S)|^4) \). Since \( \text{chase}_{e}(Q_S) = \text{owl-horst-closure}(S) \), where \( S = \text{chase}_{e-1}(Q_S) \cup R(\text{chase}_{e-1}(Q_S)) \). Since \( |S| = O(R \cup \text{chase}_{e-1}(Q_S)) \), and each member of the set \( S \) is a quad, the number of constants in \( S \), \( C(S) = 4 \times O(R \cup \text{chase}_{e-1}(Q_S)) \). Since each \( s, p, o \), such that \( c : (s, p, o) \in \text{owl-horst-closure}(S) \), is a special case of \( C(S) \),...
|\text{chase}_{i}(QS_C)| \leq |C| \cdot 64 \cdot R^2 \cdot |\text{chase}_{i-1}(QS_C)|^3|

Since |C| \leq |QS_C|, |\text{chase}_{i}(QS_C)| = O(|QS_C|^3 \cdot |\text{chase}_{i-1}(QS_C)|^3). Since i \leq |R|, |\text{chase}_{i}(QS_C)| is of the order O(|QS_C|^3 \cdot |\text{chase}_{i-1}(QS_C)|^3). 

(ii) From (i) we know that |R(\text{chase}_{i-1}(QS_C))| = O(R \cdot |\text{chase}_{i-1}(QS_C)|^3). 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: C(\text{chase}_{i+j}(QS_C)) = C(S), where S = R(\text{chase}_{i-1}(QS_C)) \cup \text{chase}_{i-1}(QS_C). Since, we already saw in (i) that |R(\text{chase}_{i-1}(QS_C))| = 4 \cdot O(R \cdot |\text{chase}_{i-1}(QS_C)|^3), |C(\text{chase}_{i+j}(QS_C))| = 4 \cdot O(R \cdot |\text{chase}_{i-1}(QS_C)|^3). Since each s, p, o, such that c(s, p, o) \in \text{chase}_{i+j}(QS_C) is from the set C(\text{chase}_{i+j}(QS_C)), |\text{chase}_{i+j}(QS_C)| \leq |C| \cdot 64 \cdot R^2 \cdot |\text{chase}_{i-1}(QS_C)|^3. Since |C| \leq |QS_C|, |\text{chase}_{i+j}(QS_C)| = O(|QS_C|^4 \cdot |\text{chase}_{i-1}(QS_C)|^3). Since i \leq |R|, |\text{chase}_{i}(QS_C)| is of the order O(|QS_C|^3 \cdot |\text{chase}_{i-1}(QS_C)|^3).

\textbf{Proposition B.1.} There exists unrestricted quad-systems whose dChase is infinite.

\textbf{Proof.} Consider an example of a quad-system QS_C = \langle Q_c, r \rangle, where graph_{Q_c}(c) = \{a, \text{rdf:type}, C\}, and the BR r = c1: (x, \text{rdf:type}, C) \rightarrow c2: (x, P, f(x)), c1: (f(x), \text{rdf:type}, C). The dChase computation starts with dChase_0(QS_C) = \{c: (a, \text{rdf:type}, C)\}, now the rule r is applicable, and its application leads to dChase_1(QS_C) = \{c: (a, \text{rdf:type}, C), c: (a, P, f(a)), c: (f(a), \text{rdf:type}, C)\}, which again is applicable for r for c: (f(a), \text{rdf:type}, C). For any i \geq 0, dChase_i(QS_C) contains c: (f(a), \text{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.

\textbf{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 \Sigma, string \vec{w} is a sequence of symbols from \Sigma. A language L is a subset of \Sigma^*. where \Sigma^* is the set of all strings that can be constructed from the alphabet \Sigma, and also includes the empty string \epsilon. 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} \rightarrow \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:

\textbf{Property B.2.} For a CFG, every PR is of the form v \rightarrow \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 \rightarrow \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 \rightarrow \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: (x_1, w_1, x_2), c: (x_2, w_2, x_3), ..., c: (x_n, w_n, x_{n+1}) \rightarrow c: (x_1, v, x_{n+1})

(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: (x, \text{rdf:type}, C) \rightarrow \exists y c: (x, t_i, y), c: (y, \text{rdf:type}, C)

(6)

and Q_c contains only the triple:

c: (a, \text{rdf:type}, C)
We in the following show that:

\[ QS_c \models \exists y c: (a, S_1, y) \land c: (a, S_2, y) \leftrightarrow L(G_1) \cap L(G_2) \neq \emptyset \quad (7) \]

Claim (1) For any \( \vec{w} = t_1, \ldots, t_p \in T^* \), there exists \( b_1, \ldots, b_p \), such that c: (a, t_1, b_1), c: (a, t_2, b_2), ..., c: (b_{p-1}, t_p, b_p), c: (b_p, 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: (a, rdf:type, C) \in dChase(QS_c), on which rules of the form (6) is applicable. Hence, there exists an \( i \) such that dChase(QS_c) contains c: (a, t_i, b_i), c: (b_i, rdf:type, C), for each \( t_i \in T \). Hence the base case.

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

**Inductive step** Suppose if \( \vec{w} = t_1 \ldots t_{p+1} \), with \(|\vec{w}| \leq p' + 1\), and for \( V_j \rightarrow^{*} \vec{w} \), there exists \( b_1, \ldots, b_{p+1} \), such that c: (a, t_1, b_1), c: (a, t_2, b_2), ..., c: (b_{p-1}, t_p, b_p), c: (b_p, rdf:type, C) \in dChase(QS_c).

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

**Claim (3)** For any \( \vec{w} = t_1 \ldots t_p \in \{V \cup T\}^* \), and for any \( V_j \in V \), if there exists \( b_1, \ldots, b_p, b_{p+1} \), with c: (a, t_1, b_1), c: (a, t_2, b_2), ..., c: (b_{p-1}, t_p, b_p), c: (b_p, rdf:type, C) \in dChase(QS_c), then \( V_j \rightarrow^{*} \vec{w} \).

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

(b) Suppose if \( QS_c \models \exists y c: (a, S_1, y) \lor c: (a, S_2, y) \), then this implies that there exists \( b_p \) such that c: (a, S_1, b_p), c: (a, S_2, b_p) \in dChase(QS_c). Then it is the case that there exists \( \vec{w} = t_1 \ldots t_p \in T^* \), and
C. Proofs of Section 4

Lemma 4.7. (i) Any c: (s, p, o) ∈ dChase(QSc) is such that s, p, o ∈ C(dChase(QSc)). Also C(dChase(QSc)) = ᦦ(i) C(dChase(QSc)) \cup B(dChase(QSc)) \cup L(dChase(QSc)), and U(dChase(QSc)) = ᦦ(QSc), L(dChase(QSc)) = L(QSc). Also B(dChase(QSc)) = B(QSc) ∪ Bsk(dChase(QSc)). Since CCharacter(QSc), LCharacter(QSc), and BCharacter(QSc) are part of the input, the set Bsk(dChase(QSc)) determines the incremental part of the Chase(QSc). Note that each b ∈ Bsk(dChase(QSc)) is a skolem blank node, whose descendence 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|C|. Also since the number of the tree is bounded by m = max{ar(fj) | fj is a skolem function occurring in sk(R)}, any such tree has at most m2|C| leaf nodes and m2|C| non-leaf nodes. Let F be the set of function skolem symbols occurring in R. Since each leaf node is a constant in C(QSc), and each non-leaf node is an element in F, the number of possible descendence trees is bounded by |F|m2|C| * |C(QSc)|m2|C|, which is triple exponential in |QSc| as |F|, m, |C|, |C(QSc)| are polynomially bounded by fixed input size |QSc|. Hence, the number of skolem blank nodes Bsk(dChase(QSc)) are finitely bounded by O(222|QSc|). Hence, |C(dChase(QSc)) is bounded by O(222|QSc|), and |dChase(QSc)| = O(222|QSc|).

(ii) From (i) |dChase(QSc)| is tris exponentially in |QSc|, and since each iteration add-at least one quadr to its dChase, the number of iterations are bounded triple exponentially in |QSc|. Also, by lemma 3.1 any iteration i can be done in time O(|dChase_{i-1}(QSc)|^|R|).

Since, using (i) |dChase_{i-1}(QSc)| = O(2^{2^{|QSc|}}), each iteration i can be done in time O(2^{(|R|+2^{|QSc|}})). Also, as number of iterations is triple exponential, computing dChase(QSc) 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 ∈ 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 C(QSc). Hence, the number of skolem blank nodes generated is O(|C(QSc)|). Hence, the set of constants in dChase(QSc) is a polynomial in |QSc|, and also is |dChase(QSc)|.

Also, since in any dChase iteration except the final one, at least one quad should be produced and the final dChase can have at most O(|QSc|^2) triples, the total number of iterations are bounded by O(|QSc|^2) (†). Since, any dChase iteration i involves only the following two operations (a) owl-horst-closure and (b) computing R(dChase_{i-1}(QSc)). (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}(QSc)|^|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 \text{PTIME}.

Lemma C.1 (Soundness). For any quad-system, QSc = (Qc, R), if the constant unsafe ∈ dChase_{safe}(QSc), then QSc is unsafe.

Proof. In order to prove the theorem, we first prove a few supporting claims. The following claim shows that any triple c: (s, p, o) with c ∈ 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 ∈ C.

Claim (1) For any quad c: (s, p, o), where c ∈ C, if c: (s, p, o) ∈ dChase_{safe}(QSc), then c: (s, p, o) ∈ dChase(QSc).

We prove this claim by induction on the number of iterations of dChase_i(QSc) and dChase_{safe}^i(QSc).

base case i = 0, trivially holds, since by construction graph_{dChase_0(QSc)}(c) = graph_{dChase_{safe}^0(QSc)}(c), for any c ∈ C.
hypothesis for any \( i \leq k, c \in C \), if \( c: (s, p, o) \in d\text{Chase}_{i}^{safe}(QS_c) \), then \( c: (s, p, o) \in d\text{Chase}_{i}(QS_c) \).

**Inductive step** If \( c: (s, p, o) \in d\text{Chase}_{k+1}^{safe}(QS_c) \) and \( c \in C \), then either (i) \( c: (s, p, o) \in d\text{Chase}_{k}^{safe}(QS_c) \) or (ii) \( c: (s, p, o) \not\in d\text{Chase}_{k}^{safe}(QS_c) \). If (i) is the case, then by our hypothesis \( c: (s, p, o) \in d\text{Chase}_k(QS_c) \), and by construction of \( d\text{Chase}_k \), \( c: (s, p, o) \in d\text{Chase}_{k+1}(QS_c) \). Else if (ii) is the case, then by construction of the safe \( d\text{Chase}_k \), there exists a quad-graph \( S \), such that for any \( c': (s', p', o') \) in \( d\text{Chase}_k^{safe}(QS_c) \) or (b) \( c': (s', p', o') \) is obtained by application of some \( r \in aug(R) \) from \( d\text{Chase}_k^{safe}(QS_c) \), and \( S \vdash_{\text{horst}} c: (s, p, o) \). If (a) is the case, then by hypothesis \( c': (s', p', o') \) in \( d\text{Chase}_k^{safe}(QS_c) \), and (b) \( c': (s', p', o') \) is obtained by application of some \( r \in aug(R) \) from \( d\text{Chase}_k^{safe}(QS_c) \), and \( S \vdash_{\text{horst}} c: (s, p, o) \). If (a) is the case, then by hypothesis \( c': (s', p', o') \) in \( d\text{Chase}_k^{safe}(QS_c) \), and (b) \( c': (s', p', o') \) is obtained by application of some \( r \in aug(R) \) from \( d\text{Chase}_k^{safe}(QS_c) \), and \( S \vdash_{\text{horst}} c: (s, p, o) \). Since any quad of the form \( c: (b', \text{descendantOf}, b) \) in \( d\text{Chase}_{k}^{safe}(QS_c) \) is not an element of \( Q_c \), and can only be introduced by an application of a BR \( r \in aug(R) \), any quad of the form \( c: (b', \text{descendantOf}, b) \) can only be introduced, earliest in the first iteration of \( d\text{Chase}_{k}^{safe}(QS_c) \). Suppose \( c_c: (b', \text{descendantOf}, b) \in d\text{Chase}_{k}^{safe}(QS_c) \), then there exists an iteration \( i \geq 1 \) such that \( c_c: (b', \text{descendantOf}, b) \in d\text{Chase}_{i}^{safe}(QS_c) \), for any \( j \geq i \). We apply induction on \( i \) for the proof.

**Base case** suppose \( c_c: (b', \text{descendantOf}, b) \in d\text{Chase}_1(QS_c) \) and since \( b \neq b' \), then there exists a BR \( r \in aug(R) \), \( \exists \mu \in M \), such that \( body(r)(\vec{x}, \vec{f}(\vec{x})) \in d\text{Chase}_1^{safe}(QS_c) \), and \( c: (b', \text{descendantOf}, b) \in body(r)(\vec{x}, \vec{f}(\vec{x})) \in d\text{Chase}_1^{safe}(QS_c) \). 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 \( b = f_i(\mu(x_1), \ldots, x_n) \). Hence \( b' \) is a descendant of \( b \) (by definition).

**Hypothesis** if \( c_c: (b', \text{descendantOf}, b) \in d\text{Chase}_i(QS_c) \), for \( 1 \leq i \leq k \), then \( b' \) is a descendant of \( b \).

**Inductive step** suppose \( c_c: (b', \text{descendantOf}, b) \in d\text{Chase}_{k+1}(QS_c) \), then either (i) \( c_c: (b', \text{descendantOf}, b) \in d\text{Chase}_k(QS_c) \) or (ii) \( c_c: (b', \text{descendantOf}, b) \not\in d\text{Chase}_k(QS_c) \). Suppose (i) is the case, then by hypothesis, \( b' \) is a descendant of \( b \). If (ii) is the case, then either (a) \( c_c: (b', \text{descendantOf}, b) \) is the result of the application of a BR \( r \in R \) on \( d\text{Chase}_k^{safe}(QS_c) \) or (b) there exists blank-node \( b'' \neq b', b' \), such that \( c_c: (b', \text{descendantOf}, b) \) is the result of application of a BR \( r \in R \) and \( c_c: (b' \text{ descendantOf}, b'') \in d\text{Chase}_k(QS_c) \). 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 \( b \).

Suppose the constant \( \text{unsafe} \in d\text{Chase}_{\text{safe}}(Q_S^c) \), then this implies that there exists an iteration \( i \) such that the function unsafeTest on \( R \) and \( d\text{Chase}_{\text{safe}}(Q_S^c) \) returns True. This implies that, there exists \( r = \text{body}(r)(\vec{x}, \vec{z}) \rightarrow \text{head}(r)(\vec{x}, \vec{f}(\vec{x})) \in \text{aug}(R), \mu \in M, b, b' \in B, f_i(\vec{x}) \in f(\vec{x}), \text{such that} body(r)(\vec{x}, \vec{z})[\mu] \in d\text{Chase}_{\text{i}}(Q_S^c), b \in \mu(\vec{x}), c_c : \{ \langle b', \text{descendantOf}, b \rangle \in d\text{Chase}_{\text{i}}(Q_S^c) \} \text{ and } \{ c | c_c : \langle b', \text{originContext}, c \rangle \in d\text{Chase}_{\text{i}}(Q_S^c) \} = \text{context}(b'_i, \text{head}(r)(\vec{x}, \vec{f}(\vec{x}))). \) 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 \( \mu \) being applied to \( r \), leads to the generation of a blank node \( b'' = f_i(\mu(\vec{x})) \), such that \( \text{context}(b'') = \text{context}(b'_i) \). Also since \( b \in \mu(\vec{x}) \) and by definition of unsafeTest, it follows that \( \{ c | c_c : \langle b, \text{originContext}, c \rangle \in d\text{Chase}_{\text{i}}(Q_S^c) \} = \text{context}(b'_i). \) By claim 2, it follows that \( \text{context}(b) = \text{context}(b'') \). Also, trivially \( b \) is a descendant of \( b'' \) (\( \triangledown \)). Also from claim 1, since \( \text{body}(r)(\vec{x}, \vec{z})[\mu] \in d\text{Chase}_{\text{i}}(Q_S^c), \text{body}(r)(\vec{x}, \vec{z})[\mu] \in d\text{Chase}_{\text{i}}(Q_S^c) \) for some iteration \( j \), and hence \( r \) is applicable on \( d\text{Chase}_{j}(Q_S^c) \) for \( \mu \), and since applying \( \mu \) on \( \text{head}(r)(\vec{x}, \vec{f}(\vec{x})) \), produces a skolem quadr in which \( b'' \) occurs, and hence \( b'' \in B_{sk}(d\text{Chase}(Q_S^c)). \) Also since \( b \in \mu(\vec{x}) \), implies that \( b \in B_{sk}(d\text{Chase}(Q_S^c)) \). Hence \( b, b' \in B_{sk}(d\text{Chase}(Q_S^c))(\bullet). \) By \( (\bullet), (\triangledown), (\triangle) \), all the prerequisites of an unsafe quad-system is satisfied, and hence \( Q_S^c \) is unsafe.

Suppose if (b) is the case, then as a result of \( \mu \) being applied to \( r \), leads to the generation of a blank node \( b'' = f_i(\mu(\vec{x})) \), such that \( \text{context}(b'') = \text{context}(b'_i) \). 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, \( c_c : \langle b', \text{descendantOf}, b \rangle \in d\text{Chase}_{\text{i}}(Q_S^c), \) by claim 3, it follows that \( b' \) is a descendant of \( b \) and by transitivity, it follows that \( b' \) is a descendant of \( b'' \) (\( \triangledown \)). Also by assumptions of unsafeTest, it follows that \( \{ c | c_c : \langle b', \text{originContext}, c \rangle \in d\text{Chase}_{\text{i}}(Q_S^c) \} = \text{context}(b''). \) By claim 2, it follows that \( \text{context}(b') = \text{context}(b'')(\bullet) \). Also from claim 1, since \( \text{body}(r)(\vec{x}, \vec{z})[\mu] \in d\text{Chase}_{\text{j}}(Q_S^c), \text{body}(r)(\vec{x}, \vec{z})[\mu] \) is also in \( d\text{Chase}_{\text{j}}(Q_S^c) \) for some iteration \( j \), and hence \( r \) is applicable on \( d\text{Chase}_{j}(Q_S^c) \) for \( \mu \), and since applying \( \mu \) on \( \text{head}(r)(\vec{x}, \vec{f}(\vec{x})) \), produces a skolem quadr in which \( b'' \) occurs, and hence \( b'' \in B_{sk}(d\text{Chase}(Q_S^c)). \) Also by claim 1, it follows that \( b, b' \in B_{sk}(d\text{Chase}(Q_S^c)). \) By definition of unsafe quad-systems and by \( (\bullet), (\triangledown), (\triangle) \), \( Q_S^c \) is unsafe.

\( \square \)

Lemma C.2 (Completeness). For any quad-system, \( Q_S^c = (Q_c, R) \), if \( Q_S^c \) is unsafe then \( \text{unsafe} \in d\text{Chase}_{\text{safe}}(Q_S^c) \).

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 \( \text{unsafe} \notin d\text{Chase}_{\text{safe}}(Q_S^c), \) then \( d\text{Chase}(Q_S^c) \subseteq d\text{Chase}_{\text{safe}}(Q_S^c) \). We approach the proof by induction on the iterations needed during the dChase computation.

(base case) since \( d\text{Chase}_0(Q_S^c) = Q_c, d\text{Chase}_0(Q_S^c) \subseteq d\text{Chase}_0(\text{safe})(Q_S^c) \).

(hypothesis) for any \( i \leq k, \) if \( \text{unsafe} \notin d\text{Chase}_i(Q_S^c), \) then \( d\text{Chase}_i(Q_S^c) \subseteq d\text{Chase}_{i}(\text{safe})(Q_S^c). \)

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

Claim (2) For any skolem blank-node \( b \) generated in \( d\text{Chase}_{\text{safe}}(Q_S^c) \), and for any \( c \in C, \) if \( c \in \text{context}(b) \), then there exists a quad \( c_c : \langle b, \text{originContext}, c \rangle \in d\text{Chase}_{\text{safe}}(Q_S^c). \)
Since, the only way a skolem blank node \( b \) gets generated in any iteration \( i \) of \( dChase_{safe}(QS_C) \), is by the application of a BR \( r \in aug(r) \), i.e. when there \( \exists r = body(r)(\vec{x}, \vec{z}) \rightarrow head(r)(\vec{x}, f(\vec{x})) \in aug(R), \exists \mu \in M, \) such that \( body(r)(\vec{x}, \vec{z})[\mu] \in dChase_{safe}^{-1}(QS_C) \), and there exists \( f_i(\vec{x}) \in \vec{f}(\vec{x}) \), with \( b = f_i(\mu(\vec{x})) \). But by construction of \( aug(r) \), \( head(r)(\vec{x}, f(\vec{x})) \) also has a quad-pattern of the form \( c_{\cdot} : (f_i(\mu(\vec{x})), \text{origin-context}, c) \), for every \( c \in cScope(f_i(\vec{x}), dChase_{safe}^{-1}(QS_C)) \), and hence also a triple of the form \( c_{\cdot} : (f_i(\mu(\vec{x})), \text{origin-context}, c) \) gets generated, on the application of \( \mu \) on \( head(r)(\vec{x}, f(\vec{x})) \). Since origin context of \( b = f_i(\mu(\vec{x})) \) is the set \( cScope(f_i(\vec{x}), dChase_{safe}^{-1}(QS_C)) \), 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 \( sub-distance(b, b') \) is defined inductively as:

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

**Claim (3)** For any two skolem blank nodes \( b, b' \) in \( dChase_{safe}(QS_C) \), if \( b \) is a descendant of \( b' \) then there exists a quad of the form \( c_{\cdot} : (b, \text{descendantOf}, b') \in dChase_{safe}(QS_C) \)

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 \( b' = f(t_1, \ldots, t_n) \) and \( b = t_i \). Since by construction, the only way \( b' \) is generated, in any iteration \( i \) of \( dChase_{safe}(QS_C) \), is by the application of a BR, i.e. when there \( \exists r = body(r)(\vec{x}, \vec{z}) \rightarrow head(r)(\vec{x}, f(\vec{x})) \in aug(R), \exists \mu \in M body(r)(\vec{x}, \vec{z})[\mu] \in dChase_{safe}^{-1}(QS_C) \), and there exists \( f_i(\vec{x}) \in \vec{f}(\vec{x}) \) with \( b' = f_i(\mu(\vec{x})) \) and \( b = \mu(x_i) \) for some \( x_i \in \vec{x} \). But by construction of \( aug(r) \), \( head(r)(\vec{x}, f(\vec{x})) \), also has a quad-pattern of the form \( c_{\cdot} : (x_i, \text{descendantOf}, f_i(\vec{x})) \). Hence application of \( \mu \) on \( head(r)(\vec{x}, f(\vec{x})) \), also produces quads of the form \( c_{\cdot} : (\mu(x_i), \text{descendantOf}, f_i(\vec{x})) \), which means that \( c_{\cdot} : (b, \text{descendantOf}, b') \in dChase_{safe}(QS_C) \).

**Hypothesis** Suppose sub-distance \( (b, b') \leq k \), for some \( 1 \leq k \in \mathbb{N} \), then \( c_{\cdot} : (b, \text{descendantOf}, b') \in dChase_{safe}(QS_C) \).

**Inductive step** Suppose sub-distance \( (b, b') = k + 1 \), then there exists a \( b'' \neq b \), such that \( b'' = f(t_1, \ldots, t_n) \), 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 \( c_{\cdot} : (b, \text{descendantOf}, b'') \in dChase_{safe}(QS_C) \), and \( c_{\cdot} : (b'', \text{descendantOf}, b') \in dChase_{safe}(QS_C) \), and since by construction \( c_{\cdot} : (\text{descendantOf}, \text{rdfs:Type}, \text{owl:TransitiveProperty}) \in dChase_{safe}(QS_C) \). Hence, \( c_{\cdot} : (b, \text{descendantOf}, b') \in dChase_{safe}(QS_C) \).

Suppose \( QS_C \) is unsafe, then by definition, there exists a blank nodes \( b, b' \) in \( B_{\cdot} k \cap dChase_{safe}(QS_C) \), such that \( b \) is descendant of \( b' \), and \( \text{originContexts}(b) = \text{originContexts}(b') \). By contradiction, if \( \text{unSafe} \not\in dChase_{safe}(QS_C) \), then by claim 1, \( dChase(QS_C) \subseteq dChase_{safe}(QS_C) \). Since by claim 2, for any \( c \in \text{originContexts}(b) \), there exists quads of the form \( c_{\cdot} : (b, \text{origin-context}, c) \in dChase_{safe}(QS_C) \) and for every \( c' \in \text{originContexts}(b') \), there exists \( c_{\cdot} : (b', \text{originContext}, c') \in dChase_{safe}(QS_C) \).

Since \( \text{originContexts}(b) = \text{originContexts}(b') \), it follows that \( \{ c \mid c_{\cdot} : (b, \text{origin-context}, c) \in dChase_{safe}(QS_C) \} = \{ c' \mid c_{\cdot} : (b', \text{origin-context}, c') \in dChase_{safe}(QS_C) \} \). Also by claim 3, since \( b \) is a descendant of \( b' \), there exists a quad of the form \( c_{\cdot} : (b, \text{descendantOf}, b') \in dChase_{safe}(QS_C) \).

But by construction of \( dChase_{safe}(QS_C) \), there should exist a \( b'' \in B_{\cdot} k \cap dChase_{safe}(QS_C) \), \( r \in aug(R), \mu \in M \), such that \( c_{\cdot} : (b, \text{descendantOf}, b'') \in dChase_{safe}(QS_C) \) and \( b'' \in \text{origin}(x_i) \), and \( b' = f_i(\mu(\vec{x})) \) and since \( \{ c \mid c_{\cdot} : (b, \text{origin-context}, c) \in dChase_{safe}(QS_C) \} = \{ c' \mid c_{\cdot} : (b', \text{origin-context}, c') \in dChase_{safe}(QS_C) \} \), the method \( \text{unSafeTest}(dChase_{safe}(QS_C), R) \) should return True, for some \( i \). Hence, it should be the case that \( \text{unSafe} \in dChase_{safe}( QS_C ) \), which is a contradiction to our assumption. Hence, \( \text{unSafe} \in dChase_{safe}(QS_C) \), if \( dChase(QS_C) \) is unsafe.

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