

Query Answering over Contextualized RDF/OWL Knowledge with Forall-Existential Bridge Rules: Decidable Finite Extension Classes

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

Mathew Joseph [a,b,*], Gabriel Kuper [b], Till Mossakowski [c], Luciano Serafini [a]
[a] *DKM, FBK-IRST, Via Sommarive 18, 38050, Trento, Italy*
[b] *DISI, University Of Trento, Via Sommarive 5, 38123, Trento, Italy*
[c] *Faculty of Computer Science, Otto-von-Guericke University of Magdeburg, 39016, Magdeburg, Germany*
E-mail: {joseph, kuper}@disi.unitn.it, {serafini}@fbk.eu, {mossakow}@iws.cs.uni-magdeburg.de

Abstract. The proliferation of contextualized knowledge in the Semantic Web (SW) has led to the popularity of knowledge formats such as *quads* in the SW community. A quad is an extension of RDF triple with the contextual information of the triple. We in this paper, study the problem of query answering over quads augmented with forall-existential bridge rules that enable interoperability of reasoning between the triples in various contexts. We call a set of quads together with such expressive bridge rules, a quad-system. Query answering over quad-systems is undecidable, in general. We derive decidable classes of quad-systems, for which query answering can be done using forward chaining. Sound, complete and terminating procedures, which are adaptations of the well known chase algorithm, are provided for these classes for deciding query entailment. Safe, msafe, and csafe class of quad-systems restrict the structure of blank nodes generated during the chase computation process to be directed acyclic graphs (DAGs) of bounded depth. RR and restricted RR classes do not allow the generation of blank nodes during the chase computation process. Both data and combined complexity of query entailment has been established for the classes derived. We further show that quad-systems are equivalent to forall-existential rules whose predicates are restricted to ternary arity, modulo polynomial time translations. We subsequently show that the technique of safety, strictly subsumes in expressivity, some of the well known and expressive techniques, such as joint acyclicity and model faithful acyclicity, used for decidability guarantees in the realm of forall-existential rules.

Keywords: Contextualized Query Answering, Contextualized RDF/OWL knowledge bases, Multi-Context Systems, Quads, Query answering, forall-existential rules, Knowledge Representation, Semantic Web

1. Introduction

As the Semantic Web (SW) is getting more and more ubiquitous and its constellation of interlinked ontologies, the web of data, is seamlessly proliferating at a steady rate, more and more applications have started using SW as a back end, providing their users manifold services, leveraging semantic technologies. One of the main reasons why SW enjoys such admirable hospitality from its mammoth geographically disparate users is its "simple" and "open" model. The model is simple, as the only intricacy that a creator/consumer of a SW

[*]Corresponding author. E-mail: joseph@disi.unitn.it

application needs to equipped with, is that of a (RDF) triple. A triple $t = (s, p, o)$, represents the most basic piece of knowledge in SW, where s called the *subject*, is an identifier for a person, place, thing, value, or a resource in general, about which the creator of t intended to express his/her knowledge using t. p called the *predicate*, is an identifier for a property, attribute, or in general a relation that relates s with the third component o, called the object, that is also an identifier for a resource, similar to s. The model is called open, as it allows anybody, anywhere around the world to freely create their RDF/OWL ontologies about a domain of their choice, and publish them in (embedded) RDF/OWL formats in their web portals, also linking via URIs to the concepts in other similarly published ontologies. Thus, the open model in order to promote reuse and freedom, imposes no arbitration mechanism for the ontologies users publish on the SW.

On the other hand, a problem caused by this open model is that an ontology which a person publishes is often his/her own perspective about a particular domain, which largely is relative to this person. As a consequence, the truth value of a piece of knowledge in the SW is context-dependent. Recently, as a solution to the aforementioned problem, SW community, adopts the use of a quad, an extension of a triple, as the primary carrier of knowledge. A quad $c\colon (s, p, o)$, thus adds the fourth component of the context c to the triple (s, p, o), thus explicating the identifier of the context in which the triple holds. As a result, more and more triple stores are becoming quad-stores. Some of the popular quad-stores are 4store[1], Openlink Virtuoso [2], and some of the current popular triple stores like Sesame[3], Allegrograph[4] internally keep track of the contexts of triples. Some of the recent initiatives in this direction have also extended existing formats like N-Triples to N-Quads, which the RDF 1.1 has introduced as a W3C recommendation. The latest Billion triple challenge datasets has been all released in the N-Quads format.

Another benefit of quads over triples are that they allow knowledge creators to specify various attributes of meta-knowledge that further qualify knowledge [2], and also allow users to query for this meta knowledge [3]. These attributes, which explicate the various assumptions under which knowledge holds, are also called *context dimensions* [4]. Examples of context dimensions are provenance, creator, intended user, creation time, validity time, geo-location, and topic. Having defined knowledge that is contextualized, as in $c_1\colon$ (Renzi, primeMinsiterOf, Italy) , one can now declare in a meta-context mc, statements such as $mc\colon (c_1$, creator, John$)$, $mc\colon (c_1$, expiryTime, "jun-2016") that talk about the knowledge in context c_1, in this case its creator and expiry time. Another benefit of such a contextualized approach is that it opens possibilities of interesting ways for querying a contextualized knowledge base. For instance, if context c_1 contains knowledge about football world cup 2014 and context c_2 about football euro cup 2012. Then the query "who beat Italy in both world cup 2014 and euro cup 2012" can be formalized as the conjunctive query:

$$c_1\colon (x, \mathsf{beat}, \mathsf{Italy}) \wedge c_2\colon (x, \mathsf{beat}, \mathsf{Italy}),$$

where x is a variable.

While reasoning with knowledge in quad form, since knowledge can be grouped and divided context wise and simultaneously be fed to separate reasoning engines, this approach improves both efficiency and scalability. Besides the above flexibility, *bridge rules* [5] can be provided for inter-operating the knowledge in different contexts. Such rules are primarily of the form:

$$c : \phi \rightarrow c' : \phi' \tag{1}$$

where ϕ, ϕ' are both atomic concept (role) symbols, c, c' are contexts. The semantics of such a rule is that if, for any $\vec{a}$, $\phi(\vec{a})$ holds in context c, then $\phi'(\vec{a})$ should hold in context c', where $\vec{a}$ is a unary/binary vector depending on whether ϕ, ϕ' are concept/role symbols. Although such bridge rules serve the purpose of specifying knowledge interoperability from a source context c to a target context c', in many practical situations there is the need of inter-operating multiple source contexts with multiple target targets, for which the bridge rules of the form (1) is inadequate. Besides, one would also want the ability of creating new values in target contexts for the bridge rules.

In this work, we study contextual reasoning and query answering over contextualized RDF/OWL knowledge in the presence of *forall-existential bridge rules* that allows conjunctions and existential quantifiers in them, and hence is more expressive than those, in DDL [5] and McCarthy et al. [6]. 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

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

distributed chase, which is an extension of the standard *chase* [19,20] that is widely used in the knowledge representation (KR) and Database (DB) settings, for similar purposes. As far as the semantics for reasoning is concerned, we adopt the approach given in works such as Distributed Description Logics [5], E-connections [21], and two-dimensional logic of contexts [22], 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. We extend the standard RDF/OWL semantics to a context-based semantics that can be used for reasoning over contextualized RDF/OWL knowledge. Studying conjunctive query answering over quad-systems. It turns out that the entailment problem of conjunctive queries is undecidable for the most general class of quad-systems, called *unrestricted quad-systems*.
2. We derive decidable subclasses of unrestricted quad-systems, namely *csafe*, *msafe*, and *safe* quad-systems, for which we detail both data and combined complexities of conjunctive query entailment. The classes are based on constrained DAG structure of Skolem blank nodes generated during the chase construction. We also provide decision procedures to decide whether an input quad-system is safe (csafe, msafe) or not.
3. We further derive less expressive classes, RR and restricted RR quad-systems, for which no Skolem blank nodes are generated during chase construction.
4. We show that the class of unrestricted quad-systems are equivalent to the class of ternary $\forall\exists$ rule sets. We compare the derived classes of quad-systems with well known subclasses of $\forall\exists$ rule sets, such as jointly acyclic and model faithful acyclic rule sets, and show that the technique of safety, we propose, subsumes these other techniques, in expressivity.

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 *csafe*, *msafe*, and *safe* quad-systems and their computational properties. In section 5, the RR quad-systems and the restricted RR quad-systems. In section 6, we prove the equivalence of quad-systems with ternary $\forall\exists$ rule sets, and formally compare a few well known decidable classes in the realm of $\forall\exists$ rules to the classes of quad-systems, we presented in section 4. We provide a detailed discussion to other relevant related works in section 7, and conclude in section 8.

Note that parts of the contents of section 2 and section 3 has been taken from conference papers [10] and [11].

2. Contextualized Quad-Systems

In this section, we formalize the notion of a quad-system and its semantics. For any vector or sequence $\vec{x}$, we denote by $\|\vec{x}\|$ the number of symbols in $\vec{x}$, and by $\{\vec{x}\}$ the set of symbols in $\vec{x}$. For any sets A and B, $A \rightarrow B$ denotes the set of all functions from set A to set B. Given the set of URIs $\mathbf{U}$, the set of blank nodes $\mathbf{B}$, and the set of literals $\mathbf{L}$, the set $\mathbf{C} = \mathbf{U} \uplus \mathbf{B} \uplus \mathbf{L}$ is called the set of (RDF) constants. Any $(s, p, o) \in \mathbf{C} \times \mathbf{C} \times \mathbf{C}$ is called a generalized RDF triple (from now on, just triple). A graph is defined as a set of triples. A *Quad* is a tuple of the form $c\colon (s, p, o)$, where (s, p, o) is a triple and c is a URI[5], called the *context identifier* that denotes the context of the RDF triple. A *quad-graph* is defined as a set of quads. For any quad-graph Q and any context identifier c, we denote by $graph_Q(c)$ the set $\{(s, p, o) | c\colon (s, p, o) \in Q\}$. We denote by $Q_{\mathcal{C}}$ the quad-graph whose set of context identifiers is $\mathcal{C}$. The set of constants occurring in $Q_{\mathcal{C}}$, given as $\mathbf{C}(Q_{\mathcal{C}}) = \{c, s, p, o \mid c\colon (s, p, o) \in Q_{\mathcal{C}}\}$. The set of URIs in $Q_{\mathcal{C}}$, is given by $\mathbf{U}(Q_{\mathcal{C}}) = \mathbf{C}(Q_{\mathcal{C}})$ $\cap$ $\mathbf{U}$. The set of blank nodes $\mathbf{B}(Q_{\mathcal{C}})$, the set of literals $\mathbf{L}(Q_{\mathcal{C}})$ are similarly defined. Let $\mathbf{V}$ be the set of variables, any element of the set $\mathbf{C}^{\mathbf{V}} = \mathbf{V} \cup \mathbf{C}$ is a *term*. Any $(s, p, o) \in \mathbf{C}^{\mathbf{V}} \times \mathbf{C}^{\mathbf{V}} \times \mathbf{C}^{\mathbf{V}}$ is called a *triple pattern*, and an expression of the form $c\colon (s, p, o)$, where (s, p, o) is a triple pattern, c a context identifier, is called a *quad pattern*. A triple pattern t, whose variables are elements of the vector $\vec{x}$ or elements of the vector $\vec{y}$ is written as $t(\vec{x}, \vec{y})$. For any function $f\colon A \rightarrow B$, the *restriction* of f to a set A', is the mapping $f|_{A'}$ from $A' \cap A$ to B s.t. $f|_{A'}(a) = f(a)$, for each $a \in A \cap A'$. For any triple pattern $t = (s, p, o)$ and a function μ from $\mathbf{V}$ to a set A, $t[\mu]$ denotes

[5] Although, in general a context identifier can be a constant, for the ease of notation, we restrict them to be a URI

$(\mu'(s), \mu'(p), \mu'(o))$, where μ' is an extension of μ to $\mathbf{C}$ s.t. $\mu'|_{\mathbf{C}}$ is the identity function. For any set of triple patterns G, $G[\mu]$ denotes $\bigcup_{t \in G} t[\mu]$. For any vector of constants $\vec{a} = \langle a_1, \ldots, a_{\|\vec{a}\|}\rangle$, and vector of variables $\vec{x}$ of the same length, $\vec{x}/\vec{a}$ is the function μ s.t. $\mu(x_i) = a_i$, for $1 \leq i \leq \|\vec{a}\|$. We use the notation $t(\vec{a}, \vec{y})$ to denote $t(\vec{x}, \vec{y})[\vec{x}/\vec{a}]$. Similarly, the above notations are also extended to sets of quad-patterns. For instance $Q(\vec{x}, \vec{y})$ denotes a set of quad-patterns, whose variables are from $\vec{x}$ or $\vec{y}$, and $Q(\vec{a}, \vec{y})$ is written for $Q(\vec{x}, \vec{y})[\vec{x}/\vec{a}]$. For the sake of interoperating knowledge in different contexts, bridge rules need to be provided:

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

$$\forall \vec{x} \forall \vec{z}\, [c_1 \colon t_1(\vec{x}, \vec{z}) \wedge \ldots \wedge c_n \colon t_n(\vec{x}, \vec{z}) \rightarrow \exists \vec{y}\, c'_1 \colon t'_1(\vec{x}, \vec{y}) \wedge \ldots \wedge c'_m \colon t'_m(\vec{x}, \vec{y})] \quad (2)$$

where $c_1, \ldots, c_n, c'_1, \ldots, c'_m$ are context identifiers, $\vec{x}$, $\vec{y}$, $\vec{z}$ are vectors of variables s.t. $\{\vec{x}\}, \{\vec{y}\}$, and $\{\vec{z}\}$ are pairwise disjoint. $t_1(\vec{x}, \vec{z})$, ..., $t_n(\vec{x}, \vec{z})$ are triple patterns which do not contain blank-nodes, and whose set of variables are from $\vec{x}$ or $\vec{z}$. $t'_1(\vec{x}, \vec{y})$, ..., $t'_m(\vec{x}, \vec{y})$ are triple patterns, whose set of variables are from $\vec{x}$ or $\vec{y}$, and also does not contain blank-nodes. For any BR r of the form (2), $body(r)$ is the set of quad patterns $\{c_1 \colon t_1(\vec{x}, \vec{z}), \ldots, c_n \colon t_n(\vec{x}, \vec{z})\}$, and $head(r)$ is the set of quad patterns $\{c'_1 \colon t'_1(\vec{x}, \vec{y}), \ldots\, c'_m \colon t'_m(\vec{x}, \vec{y})\}$, and the frontier of r, $fr(r) = \{\vec{x}\}$. Occasionally, we also note the BR r above as $body(r)(\vec{x}, \vec{z}) \rightarrow head(r)(\vec{x}, \vec{y})$. The set of terms in a BR r is:

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

The set of terms for a set of BRs R is $\mathbf{C}^{\mathbf{V}}(R) = \bigcup_{r \in R} \mathbf{C}^{\mathbf{V}}(r)$. The URIs, blank nodes, literals, variables of a BR r (resp. set of BRs R) are similarly defined, and are denoted as $\mathbf{U}(r)$, $\mathbf{B}(r)$, $\mathbf{L}(r)$, $\mathbf{V}(r)$ (resp. $\mathbf{U}(R)$, $\mathbf{B}(R)$, $\mathbf{L}(R)$, $\mathbf{V}(R)$), respectively.

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

For any quad-system, $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$, the set of constants in $QS_{\mathcal{C}}$ is given by $\mathbf{C}(QS_{\mathcal{C}}) = \mathbf{C}(Q_{\mathcal{C}}) \cup \mathbf{C}(R)$. The sets $\mathbf{U}(QS_{\mathcal{C}})$, $\mathbf{B}(QS_{\mathcal{C}})$, $\mathbf{L}(QS_{\mathcal{C}})$, and $\mathbf{V}(QS_{\mathcal{C}})$ are similarly defined for any quad-system $QS_{\mathcal{C}}$. For any quad-graph $Q_{\mathcal{C}}$ (BR r), its symbol size $\|Q_{\mathcal{C}}\|$ ($\|r\|$) is the number of symbols required to print $Q_{\mathcal{C}}$ (r). Hence, $\|Q_{\mathcal{C}}\| \approx 4 * |Q_{\mathcal{C}}|$, where $|Q_{\mathcal{C}}|$ denotes the cardinality of the set $Q_{\mathcal{C}}$. Note that $|Q_{\mathcal{C}}|$ equals the number of quads in $Q_{\mathcal{C}}$. For a BR r, $\|r\| \approx 4 * k$, where k is the number of quad-patterns in r. For a set of BRs R, $\|R\|$ is given as $\Sigma_{r \in R}\|r\|$. For any quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$, its size $\|QS_{\mathcal{C}}\| = \|Q_{\mathcal{C}}\| + \|R\|$.

Semantics In order to provide a semantics for enabling reasoning over a quad-system, we need to use a local semantics for each context to interpret the knowledge pertaining to it. Since the primary goal of this paper is a decision procedure for query answering over quad-systems based on forward chaining, we consider the following desiderata for the choice of the local semantics and its deductive machinery:

- there exists a set LIR of inference rules and an operation $\mathsf{lclosure}()$ that computes the deductive closure of a graph w.r.t to the local semantics using the inference rules in LIR,
- each inference rule in LIR is *range restricted*, i.e. non value-generating,
- given a finite graph as input, the $\mathsf{lclosure}()$ operation, terminates with a finite graph as output in polynomial time whose size is polynomial w.r.t. to the input set.

Some of the alternatives for the local semantics satisfying the above mentioned criterion are Simple, RDF, RDFS [30], OWL-Horst [26] etc. Assuming that a local semantics has been fixed, for any context c, we denote by $I^c = \langle \Delta^c, \cdot^c\rangle$ an interpretation structure for the local semantics, where Δ^c is the interpretation domain, $\cdot^c$ the corresponding interpretation function. Also $\models_{\text{local}}$ denotes the local satisfaction relation between a local interpretation structure and a graph. Given a quad graph $Q_{\mathcal{C}}$, a *distributed interpretation structure* is an indexed set $\mathcal{I}^{\mathcal{C}} = \{I^c\}_{c \in \mathcal{C}}$, where I^c is a local interpretation structure, for each $c \in \mathcal{C}$. We define the satisfaction relation $\models$ between a distributed interpretation structure $\mathcal{I}^{\mathcal{C}}$ and a quad-system $QS_{\mathcal{C}}$ as:

Definition 2.2 (Model of a Quad-System)**.** *A distributed interpretation structure $\mathcal{I}^{\mathcal{C}} = \{I^c\}_{c \in \mathcal{C}}$ satisfies a quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$, in symbols $\mathcal{I}^{\mathcal{C}} \models QS_{\mathcal{C}}$, iff all the following conditions are satisfied:*

1. *$I^c \models_{local} graph_{Q_{\mathcal{C}}}(c)$, for each $c \in \mathcal{C}$;*
2. *$a^{c_i} = a^{c_j}$, for any $a \in \mathbf{C}$, $c_i, c_j \in \mathcal{C}$;*
3. *for each BR $r \in R$ of the form (2) and for each $\sigma \in \mathbf{V} \rightarrow \Delta^{\mathcal{C}}$, where $\Delta^{\mathcal{C}} = \bigcup_{c \in \mathcal{C}} \Delta^c$, if*

$$I^{c_1} \models_{local} t_1(\vec{x}, \vec{z})[\sigma], \ldots, I^{c_n} \models_{local} t_n(\vec{x}, \vec{z})[\sigma],$$

then there exists function $\sigma' \supseteq \sigma$, s.t.

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

Condition 1 in the above definition ensures that for any model $\mathcal{I}^{\mathcal{C}}$ of a quad-graph, each $I^c \in \mathcal{I}^{\mathcal{C}}$ is a local model of the set of triples in context c. Condition 2 ensures that any constant c is rigid, i.e. represents the same resource across a quad-graph, irrespective of the context in which it occurs. Condition 3 ensures that any model of a quad-system satisfies each BR in it. Any $\mathcal{I}^{\mathcal{C}}$ s.t. $\mathcal{I}^{\mathcal{C}} \models QS_{\mathcal{C}}$ is said to be a model of $QS_{\mathcal{C}}$. A quad-system $QS_{\mathcal{C}}$ is said to be *consistent* if there exists a model $\mathcal{I}^{\mathcal{C}}$, s.t. $\mathcal{I}^{\mathcal{C}} \models QS_{\mathcal{C}}$, and otherwise said to be *inconsistent*. For any quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$, it can be the case that $graph_{Q_{\mathcal{C}}}(c)$ is locally consistent, i.e. there exists an I^c s.t. $I^c \models_{\text{local}} graph_{Q_{\mathcal{C}}}(c)$, for each $c \in \mathcal{C}$, whereas $QS_{\mathcal{C}}$ is not consistent. This is because the set of BRs R adds more knowledge to the quad-system, and restricts the set of models that satisfy the quad-system.

Definition 2.3 (Quad-system entailment)**.** *(a) A quad-system $QS_{\mathcal{C}}$ entails a quad $c\colon (s,\ p,\ o)$, in symbols $QS_{\mathcal{C}} \models c\colon (s, p, o)$, iff for any distributed interpretation structure $\mathcal{I}^{\mathcal{C}}$, if $\mathcal{I}^{\mathcal{C}} \models QS_{\mathcal{C}}$ then $\mathcal{I}^{\mathcal{C}} \models \langle\{c\colon (s, p, o)\}, \emptyset\rangle$. (b) A quad-system $QS_{\mathcal{C}}$ entails a quad-graph $Q'_{\mathcal{C}'}$, in symbols $QS_{\mathcal{C}} \models Q'_{\mathcal{C}'}$ iff $QS_{\mathcal{C}} \models c\colon (s, p, o)$ for any $c\colon (s, p, o) \in Q'_{\mathcal{C}'}$. (c) A quad-system $QS_{\mathcal{C}}$ entails a BR r iff for any $\mathcal{I}^{\mathcal{C}}$, if $\mathcal{I}^{\mathcal{C}} \models QS_{\mathcal{C}}$ then $\mathcal{I}^{\mathcal{C}} \models \langle\emptyset, \{r\}\rangle$. (d) For a set of BRs R, $QS_{\mathcal{C}} \models R$ iff $QS_{\mathcal{C}} \models r$, for every $r \in R$. (e) Finally, a quad-system $QS_{\mathcal{C}}$ entails another quad-system $QS'_{\mathcal{C}'} = \langle Q'_{\mathcal{C}'}, R'\rangle$, in symbols $QS_{\mathcal{C}} \models QS'_{\mathcal{C}'}$ iff $QS_{\mathcal{C}} \models Q'_{\mathcal{C}'}$ and $QS_{\mathcal{C}} \models R'$.*

We call the decision problems (DPs) corresponding to the entailment problems (EPs) in (a), (b), (c), (d), and (e) as *quad EP, quad-graph EP, BR EP, BRs EP, and quad-system EP*, respectively.

3. Query Answering on Quad-Systems

In the realm of quad-systems, the classical conjunctive queries or select-project-join queries are slightly extended to what we call *Contextualized Conjunctive Queries* (CCQs). A CCQ $CQ(\vec{x})$ is an expression of the form:

$$\exists \vec{y}\, q_1(\vec{x}, \vec{y}) \wedge ... \wedge q_p(\vec{x}, \vec{y}) \qquad (3)$$

where q_i, for $i = 1, ..., p$ are quad patterns over vectors of *free variables* $\vec{x}$ and *quantified variables* $\vec{y}$. A CCQ is called a boolean CCQ if it does not have any free variables. With some abuse, we sometimes discard the logical symbols in a CCQ and consider it as a set of quad-patterns. For any CCQ $CQ(\vec{x})$ and a vector $\vec{a}$ of constants s.t. $\|\vec{x}\| = \|\vec{a}\|$, $CQ(\vec{a})$ is boolean. A vector $\vec{a}$ is an *answer* for a CCQ $CQ(\vec{x})$ w.r.t. structure $\mathcal{I}^{\mathcal{C}}$, in symbols $\mathcal{I}^{\mathcal{C}} \models CQ(\vec{a})$, iff there exists assignment $\mu\colon \{\vec{y}\} \to \mathbf{B}$ s.t. $\mathcal{I}^{\mathcal{C}} \models \bigcup_{i=1,...,p} q_i(\vec{a}, \vec{y})[\mu]$. A vector $\vec{a}$ is a *certain answer* for a CCQ $CQ(\vec{x})$ over a quad-system $QS_{\mathcal{C}}$, iff $\mathcal{I}^{\mathcal{C}} \models CQ(\vec{a})$, for every model $\mathcal{I}^{\mathcal{C}}$ of $QS_{\mathcal{C}}$. Given a quad-system $QS_{\mathcal{C}}$, a CCQ $CQ(\vec{x})$, and a vector $\vec{a}$, DP of determining whether $QS_{\mathcal{C}} \models CQ(\vec{a})$ is called the *CCQ EP*. It can be noted that the other DPs over quad-systems, namely Quad/Quad-graph EP, BR(s) EP, Quad-system EP, are reducible to the CCQ EP (See property 6.6). Hence, in this paper, we primarily focus on the CCQ EP.

3.1. dChase of a Quad-System

In order to build a procedure for query answering over a quad-system, we employ what has been called in the literature, a *chase* [19,20]. Specifically, we adopt notions of the *restricted chase* in Fagin et al. [23] (also called non-oblivious chase). In order to fit the framework of quad-systems, we extend the standard notion of chase to a *distributed chase*, abbreviated *dChase*. In the following, we show how the dChase of a quad-system can be constructed.

For a set of quad-patterns S and a set of terms T, we define the relation T*-connectedness* between quad-patterns in S as follows:

- q_1 and q_2 are T-connected, if $\mathbf{C}^{\mathbf{V}}(q_1) \cap \mathbf{C}^{\mathbf{V}}(q_2) \cap T \neq \emptyset$, for any two quad-patterns q_1, $q_2 \in S$,
- if q_1 and q_2 are T-connected, and q_2 and q_3 are T-connected, then q_1 and q_3 are also T-connected, for any quad-patterns $q_1, q_2, q_3 \in S$.

It can be noted that T-connectedness is an equivalence relation and partitions S into a set of T-components (similar notion is called a *piece* in Baget et al. [14]). Note that for two distinct T-components P_1, P_2 of S, $\mathbf{C}^{\mathbf{V}}(P_1) \cap \mathbf{C}^{\mathbf{V}}(P_2) \cap T = \emptyset$. For any BR $r = body(r)(\vec{x}, \vec{z}) \to head(r)(\vec{x}, \vec{y})$, suppose $P_1, P_2, \ldots, P_k$ are the pairwise distinct $\{\vec{y}\}$-components of $head(r)(\vec{x}, \vec{y})$, then r can be replaced by the semantically equivalent set of BRs $\{body(r)(\vec{x}, \vec{z}) \to P_1, \ldots, body(r)(\vec{x}, \vec{z}) \to P_k\}$ whose symbol size is worst case quadratic w.r.t. the symbol size of r. Hence, w.l.o.g. we assume that for any BR r, the set of quad-patterns $head(r)$ is a single component w.r.t. the set of existentially quantified variables in r.

Considering the fact that the local semantics for contexts are fixed a priori (for instance RDFS), both the number of rules in the set of local inference rules LIR and the size of each rule in LIR can be assumed to be a constant. Note that each local inference rule is range restricted and do not contain existentially quantified variables in its head. Any $ir \in$ LIR is of the form:

$$\forall\vec{x}\forall\vec{z}\,[t_1(\vec{x},\vec{z}) \wedge \ldots \wedge t_k(\vec{x},\vec{z}) \rightarrow t'_1(\vec{x})],$$

where $t_i(\vec{x},\vec{z})$, for $i = 1, \ldots, n$ are triple patterns, whose variables are from $\{\vec{x}\}$ or $\{\vec{z}\}$, and $t'_1(\vec{x})$ is a triple pattern, whose variables are from $\{\vec{x}\}$. Hence, for any quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}},\, R\rangle$ in order to accomplish the effect of local inferencing in each context $c \in \mathcal{C}$, for each $ir \in$ LIR of the form (4), we could augment R with a BR ir_c of the form:

$$\forall\vec{x}\forall\vec{z}\,[c\colon t_1(\vec{x},\vec{z}) \wedge \ldots \wedge c\colon t_k(\vec{x},\vec{z}) \rightarrow c\colon t'_1(\vec{x})]$$

Since $\|$LIR$\|$ is a constant and the size of the augmentation is linear in $|\mathcal{C}|$, w.l.o.g we assume that the set R contains a BR ir_c, for each $ir \in$ LIR, $c \in \mathcal{C}$.

Given a quad-system $QS_{\mathcal{C}}$, we denote by $\mathbf{B}_{sk} \subseteq \mathbf{B}$, a set of blank nodes with unique node ids called *Skolem blank nodes*, s.t. $\mathbf{B}_{sk} \cap \mathbf{B}(QS_{\mathcal{C}}) = \emptyset$. For any BR $r = body(r)(\vec{x},\vec{z}) \rightarrow head(r)(\vec{x},\vec{y})$ and an assignment $\mu\colon \{\vec{x}\} \cup \{\vec{z}\} \rightarrow \mathbf{C}$, the *application* of μ on r is defined as:

$$apply(r,\mu) = head(r)[\mu^{ext(\vec{y})}]$$

where $\mu^{ext(\vec{y})} \supseteq \mu$ s.t. $\mu^{ext(\vec{y})}(y_i) = \_\colon b$ is a fresh blank node from $\mathbf{B}_{sk}$, for each $y_i \in \{\vec{y}\}$.

We assume that there exists an order $\prec_l$ (for instance, lexicographic order) on the set of constants. We extend $\prec_l$ to the set of quads s.t. for any two quads $c\colon (s,p,o)$ and $c'\colon (s',p',o')$, $c\colon (s,p,o) \prec_l c'\colon (s',p',o')$, iff $c \prec_l c'$, or $c = c', s \prec_l s'$, or $c = c', s = s', p \prec_l p'$, or $c = c', s = s', p = p', o \prec_l o'$. It can be noted that $\prec_l$ is a strict linear order over the set of all quads. For any quad-graph $Q_{\mathcal{C}}$, $\prec_l$-greatest quad of $Q_{\mathcal{C}}$, denoted greatestQuad$_{\prec_l}(Q_{\mathcal{C}})$, is the quad $q \in Q_{\mathcal{C}}$ s.t. $q' \prec_l q$, for every other $q' \in Q_{\mathcal{C}}$. Also, the order $\prec_q$ is defined over the set of quad-graphs as follows: for any two quad-graphs $Q_{\mathcal{C}}$, $Q'_{\mathcal{C}'}$,

$Q_{\mathcal{C}} \prec_q Q'_{\mathcal{C}'}$, if (i) $Q_{\mathcal{C}} \subset Q'_{\mathcal{C}'}$;
$Q_{\mathcal{C}} \prec_q Q'_{\mathcal{C}'}$, if (i) does not hold and (ii) greatestQuad$_{\prec_l}(Q_{\mathcal{C}} \setminus Q'_{\mathcal{C}'}) \prec_l$ greatestQuad$_{\prec_l}(Q'_{\mathcal{C}'} \setminus Q_{\mathcal{C}})$;
$Q_{\mathcal{C}} \not\prec_q Q'_{\mathcal{C}'}$, if both (i) and (ii) are not satisfied;

A relation R over a set A is called a *strict linear order* iff R is irreflexive, transitive, and $R(a,b)$ or $R(b,a)$ holds, for every distinct $a, b \in A$.

Property 3.1. *Let $\mathcal{Q}$ be the set of all quad-graphs; $\prec_q$ is a strict linear order over $\mathcal{Q}$.*

Also, we define the level of a quad in dChase of a quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}},\, R\rangle$ as follows: any quad in $Q_{\mathcal{C}}$ is of level 0. The level of a set of quads is the largest among levels of quads in the set. Level of any quad that results from the application of a BR r w.r.t. an assignment μ is one more than the level of the set $body(r)[\mu]$, if it has already not been assigned a level. Let $\prec$ be an ordering on the quad-graphs s.t. for any two quad-graphs $Q'_{\mathcal{C}'}$ and $Q''_{\mathcal{C}''}$ of the same level, $Q'_{\mathcal{C}'} \prec Q''_{\mathcal{C}''}$, iff $Q'_{\mathcal{C}'} \prec_q Q''_{\mathcal{C}''}$. For $Q'_{\mathcal{C}'}$ and $Q''_{\mathcal{C}''}$ of different levels, $Q'_{\mathcal{C}'} \prec Q''_{\mathcal{C}''}$, iff level of $Q'_{\mathcal{C}'}$ is less than level of $Q''_{\mathcal{C}''}$. It can easily be seen that $\prec$ is a strict linear order over the set of quad-graphs. For any BRs r, r' and assignments μ, μ' over $\mathbf{V}(body(r)), \mathbf{V}(body(r'))$, respectively, $(r,\mu) \prec (r',\mu')$ iff $body(r)[\mu] \prec body(r')[\mu']$. For any quad-graph $Q'_{\mathcal{C}'}$, a set of BRs R, a BR $r \in R$, an assignment $\mu \in \mathbf{V}(body(r)) \rightarrow \mathbf{C}$, the boolean function $applicable_R(r,\mu,Q'_{\mathcal{C}'})$ is defined as:

$True$, if (a) $body(r)[\mu] \subseteq Q'_{\mathcal{C}'}, head(r)[\mu''] \not\subseteq Q'_{\mathcal{C}'}$, $\forall\mu'' \supseteq \mu$, and (b) $\nexists r' \in R, \nexists\mu' s.t.\ r' \neq r$ or $\mu' \neq \mu$ with $(r',\mu') \prec (r,\mu)$ and $applicable_R(r',\mu',Q'_{\mathcal{C}'})$;
$False$, otherwise;

For any quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$, let
$dChase_0(QS_{\mathcal{C}}) = Q_{\mathcal{C}}$;
$dChase_{i+1}(QS_{\mathcal{C}}) = dChase_i(QS_{\mathcal{C}}) \cup apply(r,\mu)$, if there exists $r = body(r)(\vec{x},\vec{z}) \rightarrow head(r)(\vec{x},\vec{y}) \in R$, assignment $\mu\colon \{\vec{x}\} \cup \{\vec{z}\} \rightarrow \mathbf{C}$ s.t. $applicable_R(r,\mu, dChase_i(QS_{\mathcal{C}}))$;
$dChase_{i+1}(QS_{\mathcal{C}}) = dChase_i(QS_{\mathcal{C}})$, otherwise;
for any $i \in \mathbb{N}$. The dChase of $QS_{\mathcal{C}}$, noted $dChase(QS_{\mathcal{C}})$, is given as:

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

Intuitively, $dChase_i(QS_{\mathcal{C}})$ can be thought of as the state of $dChase(QS_{\mathcal{C}})$ at the end of iteration i. It can be noted that, if there exists i s.t. $dChase_i(QS_{\mathcal{C}}) = dChase_{i+1}(QS_{\mathcal{C}})$, then $dChase(QS_{\mathcal{C}}) = dChase_i(QS_{\mathcal{C}})$. The dChase $dChase(QS_{\mathcal{C}})$ of a consistent quad-system $QS_{\mathcal{C}}$ is a *universal model* [29] of the quad-system, i.e. it is a model of $QS_{\mathcal{C}}$, and for any model $\mathcal{I}^{\mathcal{C}}$ of $QS_{\mathcal{C}}$, there is a homomorphism from $dChase(QS_{\mathcal{C}})$ to $\mathcal{I}^{\mathcal{C}}$. Hence, for any boolean CCQ $CQ()$, $QS_{\mathcal{C}} \models CQ()$ iff there exists a map

$\mu\colon \mathbf{V}(CQ) \to \mathbf{C}$ s.t. $\{CQ()\}[\mu] \subseteq dChase(QS_{\mathcal{C}})$. We call the sequence $dChase_0(QS_{\mathcal{C}}), dChase_1(QS_{\mathcal{C}})$, ..., the *dChase sequence* of $QS_{\mathcal{C}}$. The following lemma shows that in a dChase sequence of a quad-system, any dChase iteration can be performed in time exponential w.r.t the size of the largest BR.

Lemma 3.2. *For a quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$, for any $i \in \mathbb{N}^+$, the following holds: (i) $dChase_i(QS_{\mathcal{C}})$ can be computed in time $\mathcal{O}(\,|R|{*}\|dChase_{i-1}(QS_{\mathcal{C}})\|^{rs})$, where $rs = max_{r\in R}\|r\|$. (ii) $\|dChase_i(QS_{\mathcal{C}})\| = \mathcal{O}(\|dChase_{i-1}(QS_{\mathcal{C}})\| + \|R\|)$.*

Proof. (i) We can first find, if there exists an $r \in R$, assignment μ s.t. $applicable_R(r,\, \mu,\, dChase_{i-1}(QS_{\mathcal{C}}))$ holds, in the following naive way: (1) bind the set of variables in all rules in R with the set of constants in $dChase_{i-1}(QS_{\mathcal{C}})$. Let this set be called S. Note that $|S| = \mathcal{O}(|R| * \|dChase_{i-1}(QS_{\mathcal{C}}\ )\|^{\|rs\|})$, where $rs = max_{r\in R}\|r\|$. Also, note that each of the binding in S is of the form $body(r)(\vec{x},\, \vec{z})(\mu) \to head(r)(\vec{x},\, \vec{y})(\mu')$ ($\heartsuit$), where $r \in R$. (2) From the set S we filter out every binding of the form ($\heartsuit$) in which $\vec{x}[\mu] \neq \vec{x}[\mu']$. Let S' be the resulting set after the above filtering operation. (3) From the set S', we now filter out all the bindings of the form ($\heartsuit$) with $head(r)(\vec{x},\, \vec{y})(\mu') \subseteq dChase_{i-1}(QS_{\mathcal{C}})$, with resulting set S''. (4) If $S'' = \emptyset$, then there no $r \in R$, assignment μ s.t. $applicable_R(r,\, \mu,\, dChase_{i-1}(QS_{\mathcal{C}}))$ is True. Otherwise if $S'' \neq \emptyset$, then note that each binding of the form ($\heartsuit$) in S'' is s.t. condition (a) of the true $applicable_R(r,\, \mu,\, dChase_{i-1}(QS_{\mathcal{C}}))$ is satisfied. Now, we can sort S'' w.r.t. $\prec$ and select the least binding b of the form ($\heartsuit$), so that condition (b) in True condition of $applicable_R()$ is satisfied for b. It can easily seen that $applicable_R(r,\, \mu,\, dChase_{i-1}(QS_{\mathcal{C}}))$ holds for the r, μ extracted from b. Since, the size of each binding is at most $\|rs\|$, the operations (1)-(4) can be performed in time $\mathcal{O}(|R| * \|dChase_{i-1}(QS_{\mathcal{C}})\|^{rs})$. Since $dChase_i(QS_{\mathcal{C}}) = dChase_{i-1}(QS_{\mathcal{C}}) \cup head(r)[\mu]$, for r, μ with $applicable_R(r,\, \mu,\, dChase_{i-1}(QS_{\mathcal{C}}))$, $dChase_i(QS_{\mathcal{C}})$ can be computed in time $\mathcal{O}(\|dChase_{i-1}(QS_{\mathcal{C}}\ )\|^{rs})$.

(ii) Trivially holds, since in the worst case $dChase_i(QS_{\mathcal{C}}) = dChase_{i-1}(QS_{\mathcal{C}}) \cup head(r)[\mu]$, for $r \in R$. $\square$

Lemma 3.3. *For any quad-system $QS_{\mathcal{C}}$, If $\_\colon b$ is a Skolem blank node in $dChase(QS_{\mathcal{C}})$, generated by the application of assignment μ on $r = body(r)(\vec{x},\, \vec{z}) \to head(r)(\vec{x},\, \vec{y})$, with $\mu^{ext(\vec{y})}(y_j) = \_\colon b$, $y_j \in \{\vec{y}\}$, then $\_\colon b$ is unique for $(r, y_j, \vec{x}[\mu^{ext(\vec{y})}])$.*

Proof. By contradiction, suppose if $\_\colon b$ is not unique for $(r, y_j, \vec{x}[\mu^{ext(\vec{y})}])$, i.e. there exists $\_\colon b' \neq \_\colon b$ in $dChase(QS_{\mathcal{C}})$, with $\_\colon b'$ generated by r s.t. $\_\colon b' = \mu'^{ext(\vec{y})}(y_j)$ and $\vec{x}[\mu^{ext(\vec{y})}] = \vec{x}[\mu'^{ext(\vec{y})}]$. W.l.o.g. suppose $\_\colon b$ was generated in an iteration $l \in \mathbb{N}$ and $\_\colon b'$ in an iteration $m > l$. This means that $head(r)(\vec{x},\, \vec{y})[\mu^{ext(\vec{y})}] \subseteq dChase_l(QS_{\mathcal{C}})$ and hence, $head(r)(\vec{x},\, \vec{y})[\mu^{ext(\vec{y})}] \subseteq dChase_{m-1}(QS_{\mathcal{C}})$. This means that $applicable_R(r,\, \mu',\, dChase_{m-1}(QS_{\mathcal{C}}))$ is false, as $\mu' = \mu$, and our assumption that $\_\colon b' = y_j[\mu'^{ext(\vec{y})}]$ is false. Hence, $\_\colon b$ is unique for $(r,\, y_j,\, \vec{x}[\mu^{ext(\vec{y})}])$. $\square$

Although, we now know how to compute the dChase of a quad-system, which can be used for deciding CCQ EP, the following proposition reveals that for the class of quad-systems whose BRs are of the form (2), which we call *unrestricted quad-systems*, the dChase can be infinite.

Proposition 3.4. *There exists unrestricted quad-systems whose dChase is infinite.*

Proof. Consider an example of a quad-system $QS_c = \langle Q_c, r\rangle$, where $Q_c = \{c\colon (a,\, \texttt{rdf:type},\, C)\}$, and the BR $r = c\colon (x,\, \texttt{rdf:type},\, C) \to \exists y\ c\colon (x,\, P,\, y),\ c\colon (y,\, \texttt{rdf:type},\, C)$. The dChase computation starts with $dChase_0(QS_c) = \{c\colon (a,\, \texttt{rdf:type},\, C)\}$, now the rule r is applicable, and its application leads to $dChase_1(QS_c) = \{c\colon (a,\, \texttt{rdf:type},\, C),\ c\colon (a, P, \_\colon b_1),\ c\colon (\_\colon b_1,\, \texttt{rdf:type},\, C)\}$, where $\_\colon b_1$ is a fresh Skolem blank node. It can be noted that r is yet again applicable on $dChase_1(QS_c)$, for $c\colon (\_\colon b_1,\, \texttt{rdf:type},\, C)$, which leads to the generation of another Skolem blank node, and so on. Hence, $dChase(QS_c)$ does not have a finite fix-point, and $dChase(QS_c)$ is infinite. $\square$

A class $\mathfrak{C}$ of quad-systems is called a *finite extension class* (FEC), iff every member $QS_{\mathcal{C}} \in \mathfrak{C}$, $dChase(QS_{\mathcal{C}})$ is a finite set. Therefore, the class of unrestricted quad-systems is not a FEC. This raises the question if there are other approaches that can be used, for instance, a similar problem of non-finite chase is manifested in description logics (DLs) with value creation, due to the presence of existential quantifiers, whereas the approaches like the one in Glimm et al. [27] provides an algorithm for CQ entailment based on query rewriting. The theorem 3.5 below establishes the fact that the CCQ EP for unrestricted quad-systems is undecidable. Despite this, the reader should note that the following undecidability result and its proof is only provided for the sake of self containedness, and we do not claim the undecidability theorem nor its proof to be a novel contribution, as we will show in section 6, ternary $\forall\exists$ rule sets are polynomially reducible to unrestricted quad-systems. Hence, the undecidability results provided in Baget et al. [14] can trivially be applied in our setting to obtain the undecidability result for unrestricted quad-systems.

Theorem 3.5. *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 s.t. $T \cap (V_1 \cup V_2) = \emptyset$ is the set of terminals. $S_1 \in V_1$ is the start symbol of G_1, and P_1 are the set of PRs of the form $v \to \vec{w}$, where $v \in V$, $\vec{w}$ is a sequence of the form $w_1...w_n$, where $w_i \in V_1 \cup T$. Similarly s_2, P_2 is defined. Deciding whether the language generated by the grammars $L(G_1)$ and $L(G_2)$ have non-empty intersection is known to be undecidable [32].

Given two CFGs $G_1 = \langle V_1, T, S_1, P_1\rangle$ and $G_2 = \langle V_2, T, S_2, P_2\rangle$, we encode grammars G_1, G_2 into a quad-system $QS_c = \langle Q_c, R\rangle$, with only a single context identifier c. Each PR $r = v \to \vec{w} \in P_1 \cup P_2$, with $\vec{w} = w_1 w_2 w_3 .. w_n$, is encoded as a BR of the form: $c\colon (x_1, w_1, x_2), c\colon (x_2, w_2, x_3), ..., c\colon (x_n, w_n, x_{n+1}) \to c\colon (x_1, v, x_{n+1})$, where $x_1, .., x_{n+1}$ are variables. For each terminal symbol $t_i \in T$, R contains a BR of the form: $c\colon (x, \texttt{rdf:type}, C) \to \exists y \quad c\colon (x, t_i, y)$, $c\colon (y, \texttt{rdf:type}, C)$ and Q_c is the singleton: $\{ c\colon (a, \texttt{rdf:type}, C)\}$. It can be observed that:

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

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

4. Safe, Msafe and Csafe Quad-Systems: Decidable FECs

In the previous section, we saw that the query answering problem over unrestricted quad-systems is undecidable, in general. We will also see in section 6 that any quad-system is polynomially translatable to a $\forall\exists$ rule set, which is also a first order logic theory. Hence, a possible solution approach is to translate to these more expressive languages, and apply well known tests (see related work for details on such tests) available in these languages to check if query answering is decidable. If the translated quad-system passes one of these tests, then query answering can be performed on this translation using available algorithms in these expressive languages. But, such an approach is often discouraged, because of the non-applicability of the already available tools and techniques available for reasoning over quads. Instead, we in the following define three classes of quad-systems, namely SAFE, MSAFE and CSAFE, that are FECs and for which query entailment is decidable. Finiteness/Decidability is achieved by putting certain restrictions (explained below) on the blank nodes generated in the dChase.

Recall that, for any quad-system $QS_\mathcal{C}$ the set of blank-nodes $\mathbf{B}(dChase(QS_\mathcal{C}))$ in its $dChase(QS_\mathcal{C})$ not only contain blank nodes present in $QS_\mathcal{C}$ i.e. $\mathbf{B}(QS_\mathcal{C})$, but also contain Skolem blank nodes that are generated during the dChase construction process. Note that the following relation holds: $\mathbf{B}_{sk}(dChase(QS_\mathcal{C})) = \mathbf{B}(dChase(QS_\mathcal{C})) \setminus \mathbf{B}(QS_\mathcal{C})$. We assume w.l.o.g. that for any set of BRs R, any BR in R has a unique rule identifier, and we often write r_i for the BR in R, whose identifier is i.

Definition 4.1 (Origin RuleId/Vector)**.** *For any Skolem blank node* $\_\colon b$*, generated in the dChase by the application of a BR* $r_i = body(r_i)(\vec{x}, \vec{z}) \to head(r_i)(\vec{x}, \vec{y})$ *using assignment* $\mu\colon \{\vec{x}\} \cup \{\vec{z}\} \to \mathbf{C}$*, i.e.* $\_\colon b = \mu^{ext(\vec{y})}(y_j)$*, for some* $y_j \in \vec{y}$*, with* $\vec{x}[\mu^{ext(\vec{y})}] = \vec{w}$*, we say the origin ruleId (resp. vector) of* $\_\colon b$ *is* i *(resp.* $\vec{w}$*), noted* $originRuleId(\_\colon b) = i$ *(resp.* $originVector(\_\colon b) = \vec{w}$*).*

As we saw in lemma 3.3, any such Skolem blank node $\_\colon b$, generated in the dChase can uniquely be represented by the expression $(i, j, \vec{w})$, where i is rule id, j is identifier of the existentially quantified variable y_j in r_i substituted by $\_\colon b$ during the application of μ on r_i. Also in the above case, we denote relation between each constant $k = \mu^{ext(\vec{y})}(x_h)$, $x_h \in \{\vec{x}\}$, to $\_\colon b$ with the relation *childOf*. Moreover, since children of a Skolem blank node can be Skolem blank nodes, which themselves can have children, one can naturally define relation *descendantOf*=childOf$^+$ as the transitive closure of childOf. Note that according to the above definition, 'descendantOf' is not reflexive. In addition, we could keep track of the set of contexts in which a blank-node was first generated, using the following notion:

Definition 4.2 (Origin-contexts)**.** *For any quad-system $QS_{\mathcal{C}}$ and for any Skolem blank node $\_:b \in \mathbf{B}_{sk}(dChase(QS_{\mathcal{C}}))$, the set of origin-contexts of $\_:b$ is given by $originContexts(\_:b) = \{c \mid \exists i.\ c{:}(s,p,o) \in dChase_i(QS_{\mathcal{C}})$, $s = \_:b$ or $p = \_:b$ or $o = \_:b$, and $\nexists j < i$ with $c'{:}(s',p',o') \in dChase_j(QS_{\mathcal{C}})$, $s' = \_:b$ or $p' = \_:b$ or $o' = \_:b$, for any $c' \in \mathcal{C}\}$.*

Intuitively, origin-contexts for a Skolem blank node $\_:b$ is the set of contexts in which triples containing $\_:b$ are first generated, during the dChase construction. Note that there can be multiple contexts to which $\_:b$ can simultaneously be generated. By setting $originRuleId(k) = n.d.$, (resp. $originVector(k) = n.d.$, resp. $originContexts(k) = n.d.$,) where $n.d.$ is an ad hoc constant, for every $k \notin \mathbf{B}_{sk}(dChase(QS_{\mathcal{C}}))$, we extend the definition of origin ruleId, (resp. origin vector, resp. origin-contexts) to all the constants in the dChase of a quad-system.

Example 4.3. Consider the quad-system $\langle Q_{\mathcal{C}}, R\rangle$, where $Q_{\mathcal{C}} = \{c_1{:}\ (a,b,c)\}$. Suppose R is the following set:

$$R = \left\{ \begin{array}{ll} c_1{:}\ (x_{11},x_{12},z_1) \rightarrow c_2{:}\ (x_{11},x_{12},y_1) & (r_1) \\ c_2{:}\ (z_{21},z_{22},x_2) \rightarrow c_3{:}\ (y_{21},y_{22},x_2) & (r_2) \\ c_3{:}\ (z_3,x_{31},x_{32}) \rightarrow c_2{:}\ (y_3,x_{31},x_{32}) & (r_3) \end{array} \right\}$$

Suppose that for brevity quantifiers have been omitted, and variables of the form y_i or y_{ij} are implicitly existentially quantified. Iterations during dChase construction are:

$$dChase_0(QS_{\mathcal{C}}) = \{c_1{:}(a,b,c)\}$$

$$dChase_1(QS_{\mathcal{C}}) = \{c_1{:}\ (a,b,c), c_2{:}\ (a,b,\_:b_1)\}$$

$$dChase_2(QS_{\mathcal{C}}) = \{c_1{:}(a,b,c), c_2{:}\ (a,b,\_:b_1), c_3{:}\ (\_:b_2,\_:b_3,\_:b_1)\}$$

$$dChase_3(QS_{\mathcal{C}}) = \{c_1{:}(a,b,c), c_2(a,b,\_:b_1), c_3{:}\ (\_:b_2,\_:b_3,\_:b_1), c_2{:}\ (\_:b_4,\_:b_3,\_:b_1)\}$$

$$dChase_4(QS_{\mathcal{C}}) = dChase_3(QS_{\mathcal{C}}),$$

Also note:
$originRuleId(\_:b_1) = 1$, $originRuleId(\_:b_2) = originRuleId(\_:b_3) = 2$, $originRuleId(\_:b_4) = 3$,
$originVector(\_:b_1) = \langle a,\ b\rangle$, $originVector(\_:b_2) = originVector(\_:b_3) = \langle \_:b_1\rangle$, $originVector(\_:b_4) = \langle \_:b_3, \_:b_1\rangle$,

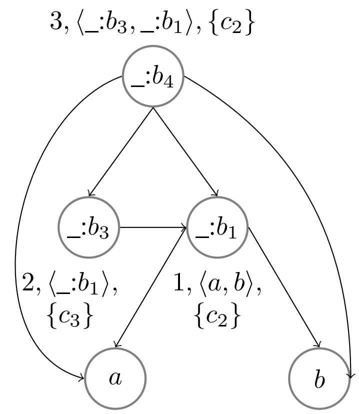

Fig. 1. descendance graph of $\_:b_4$ in example 4.3. Note: n.d. labels note shown

$originContexts(\_:b_1) = \{c_2\}$, $originContexts(\_:b_2) = originContexts(\_:b_3) = \{c_3\}$, $originContexts(\_:b_4) = \{c_2\}$,
Also $\_:b_1$ descendantOf $\_:b_3$, $\_:b_1$ descendantOf $\_:b_2$, $\_:b_3$ descendantOf $\_:b_4$, $\_:b_1$ descendantOf $\_:b_4$.

For any Skolem blank node $\_:b$ (in dChase), its descendant hierarchy can be analyzed using a *descendance graph* $\langle V, E, \lambda_r, \lambda_v, \lambda_c\rangle$, which is a labeled graph rooted at $\_:b$, whose set of nodes V are constants in the dChase, the set of edges E is s.t. $(k,k') \in E$, iff k' is a descendant of k. λ_r, λ_v, λ_c are node labeling functions $\lambda_r(k) = originRuleId(k)$, $\lambda_v(k) = originVector(k)$, s.t. $\lambda_c(k) = originContexts(k)$, for any $k \in V$. Descendance graph for $\_:b_4$ of example 4.3 is shown in Fig.1. For any two vectors of constants $\vec{v}, \vec{w}$, we note $\vec{v} \cong \vec{w}$, iff there exists a bijection $\mu : \mathbf{B}(\vec{v}) \rightarrow \mathbf{B}(\vec{w})$ s.t. $\vec{w} = \vec{v}[\mu]$.

Definition 4.4 (safe, msafe, csafe quad-systems)**.** *A quad-system $QS_{\mathcal{C}}$ is said to be* unsafe *(resp.* unmsafe*, resp.* uncsafe*), iff there exists Skolem blank nodes $\_:b \neq \_:b'$ in $dChase(QS_{\mathcal{C}})$ s.t. $\_:b$ is a descendant of $\_:b'$, with $originRuleId(\_:b) = originRuleId(\_:b')$ and $originVector(\_:b) \cong originVector(\_:b')$ (resp. $originRuleId(\_:b) = originRuleId(\_:b')$, resp. $originContexts(\_:b) = originContexts(\_:b')$). A quad-system is* safe *(*msafe*,* csafe*) iff it is not unsafe (resp. unmsafe, resp. uncsafe).*

Intuitively, safe, msafe and csafe quad-systems, does not allow repetitive generation of Skolem blank-nodes with a certain set of attributes in its dChase. The containment relation between the class of safe, msafe, and csafe quad-systems are established by the following theorem:

Theorem 4.5. *Let* SAFE, MSAFE, *and* CSAFE *denote the class of safe, msafe, and csafe quad-systems, respectively, then the following holds:*

$$\text{CSAFE} \subset \text{MSAFE} \subset \text{SAFE}$$

Proof. We first show MSAFE $\subseteq$ SAFE, by showing the inverse inclusion of their compliments, i.e. UNSAFE $\subseteq$ UNMSAFE. Suppose a given quad-system $QS_{\mathcal{C}}$ is unsafe, then by definition its dChase contains two distinct Skolem blank nodes $\_\!:b$, $\_\!:b'$ s.t. $\_\!:b$ is a descendant of $\_\!:b'$, with $originRuleId(\ \_\!:b) = originRuleId(\ \_\!:b')$ and $originVector(\ \_\!:b) \cong originVector(\ \_\!:b')$ But this implies that $originRuleId(\ \_\!:b) = originRuleId(\ \_\!:b')$. Hence, by definition $QS_{\mathcal{C}}$ is unmsafe. Hence, UNSAFE $\subseteq$ UNMSAFE (†)

Now, we show that CSAFE $\subseteq$ MSAFE, by showing that UNMSAFE $\subseteq$ UNCSAFE. Suppose a given quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$ is unmsafe, then by definition its dChase contains two distinct Skolem blank nodes $\_\!:b$, $\_\!:b'$ s.t. $\_\!:b$ is a descendant of $\_\!:b'$, with $originRuleId(\_\!:b) = originRuleId(\_\!:b')$. But this implies that there exists a BR $r_i = body(r_i)(\vec{x}, \vec{z}) \rightarrow head(r_i)(\vec{x}, \vec{y})$, assignment μ, (resp. μ',) s.t. $\_\!:b$ (resp. $\_\!:b'$) was generated in $dChase(QS_{\mathcal{C}})$ as result of application of μ (resp. μ') on r_i. That is $\_\!:b = y_j[\mu^{ext(\vec{y})}]$, and $\_\!:b' = y_k[\mu'^{ext(\vec{y})}]$, where $y_j, y_k \in \vec{y}$. We have the following two subcases (i) $j = k$, (ii) $j \neq k$: Suppose $j = k$, then it immediately follows that $originContexts(\_\!:b) = originContexts(\_\!:b')$. Hence, $QS_{\mathcal{C}}$ is uncsafe. Suppose $j \neq k$, then by construction of dChase, on application of μ' on r_i, along with $\_\!:b'$, there gets also generated a Skolem blank node $\_\!:b'' = y_j[\mu'^{ext(\vec{y})}]$, with $y_j \in \vec{y}$. Since, $\_\!:b$ and $\_\!:b''$ are generated by substitutions of the same variable $y_j \in \vec{y}$ of BR r_i, $originContexts(\_\!:b) = originContexts(\_\!:b'')$. Also since childOf($\_\!:b'$) = childOf($\_\!:b''$) $= \{\vec{x}[\mu'^{ext(\vec{y})}]\}$, $\_\!:b$ is a descendant of $\_\!:b''$. Hence, by definition, it holds that $QS_{\mathcal{C}}$ is uncsafe. Hence, UNMSAFE $\subseteq$ UNCSAFE (‡).

From † and ‡, it follows that CSAFE $\subseteq$ MSAFE $\subseteq$ SAFE. To show that the containments are strict, consider the quad-system $QS_{\mathcal{C}}$ in example 4.3. By definition, $QS_{\mathcal{C}}$ is msafe, however uncsafe, as the Skolem blank nodes $\_\!:b_1$, $\_\!:b_4$, which have the same origin contexts are s.t. $\_\!:b_1$ is a descendant of $\_\!:b_4$. Hence, CSAFE $\subset$ MSAFE. For MSAFE $\subset$ SAFE, the following example shows an instance of a quad-system that is unmsafe, yet is safe. □

Example 4.6. where $Q_{\mathcal{C}} = \{c_1\colon (a,b,c), c_2\colon (c,d,e)\}$ R is given by:

$$c_1\colon (x_{11}, x_{12}, x_{13}), c_2\colon (x_{13}, x_{14}, z_1) \rightarrow c_3\colon (y_1, x_{11}, x_{12}), c_4\colon (x_{12}, x_{13}, x_{14}) \quad (r_1)$$

$$c_3\colon (x_{21}, a, x_{22}), c_4\colon (x_{22}, x_{23}, x_{24}) \rightarrow c_1\colon (x_{21}, a, x_{22}), c_2\colon (x_{22}, x_{23}, x_{24}) \quad (r_2)$$

$$c_3\colon (x_{21}, x_{22}, a), c_4\colon (a, x_{23}, x_{24}) \rightarrow c_1\colon (x_{21}, x_{22}, a), c_2\colon (a, x_{23}, x_{24}) \quad (r_3)$$

$$c_3\colon (x_{21}, x_{22}, x_{23}), c_4\colon (x_{23}, a, x_{24}) \rightarrow c_1\colon (x_{21}, x_{22}, x_{23}), c_2\colon (x_{23}, a, x_{24}) \quad (r_4)$$

$$c_3\colon (x_{21}, x_{22}, x_{23}), c_4\colon (x_{23}, x_{24}, a) \rightarrow c_1\colon (x_{21}, x_{22}, x_{23}), c_2\colon (x_{23}, x_{24}, a) \quad (r_5)$$

Note that for brevity quantifiers have been omitted, and variables of the form y_i or y_{ij} are implicitly existentially quantified. Iterations during dChase construction are:

$$dChase_0(QS_{\mathcal{C}}) = \{c_1{:}(a,b,c), c_2{:}(c,d,e)\}$$

$$dChase_1(QS_{\mathcal{C}}) = dChase_0(QS_{\mathcal{C}}) \cup \{c_3\colon (\_\!:b_1, a, b), c_4\colon (b,c,d)\}$$

$$dChase_2(QS_{\mathcal{C}}) = dChase_1(QS_{\mathcal{C}}) \cup \{c_1\colon (\_\!:b_1, a, b), c_2\colon (b,c,d)\}$$

$$dChase_3(QS_{\mathcal{C}}) = dChase_2(QS_{\mathcal{C}}) \cup \{c_3\colon (\_\!:b_2, \_\!:b_1, a), c_4\colon (a,b,c)\}$$

$$dChase_4(QS_{\mathcal{C}}) = dChase_3(QS_{\mathcal{C}}) \cup \{c_1\colon (\_\!:b_2, \_\!:b_1, a), c_2\colon (a,b,c)\}$$

$$dChase_5(QS_{\mathcal{C}}) = dChase_4(QS_{\mathcal{C}}) \cup \{c_3\colon (\_\!:b_3, \_\!:b_2, \_\!:b_1), c_4\colon (\_\!:b_1, a, b)\}$$

$$dChase_6(QS_{\mathcal{C}}) = dChase_5(QS_{\mathcal{C}}) \cup \{c_1\colon (\_\!:b_3, \_\!:b_2, \_\!:b_1), c_2\colon (\_\!:b_1, a, b)\}$$

$$dChase_7(QS_{\mathcal{C}}) = dChase_6(QS_{\mathcal{C}}) \cup \{c_3\colon (\_\!:b_4, \_\!:b_3, \_\!:b_2), c_4\colon (\_\!:b_2, \_\!:b_1, a)\}$$

$$dChase_8(QS_{\mathcal{C}}) = dChase_7(QS_{\mathcal{C}}) \cup \{c_1\colon (\_\!:b_4, \_\!:b_3, \_\!:b_2), c_2\colon (\_\!:b_2, \_\!:b_1, a)\}$$

$$dChase_9(QS_{\mathcal{C}}) = dChase_8(QS_{\mathcal{C}}) \cup \{c_3\colon (\_\!:b_5, \_\!:b_4, \_\!:b_3), c_4\colon (\_\!:b_3, \_\!:b_2, \_\!:b_1)\}$$

$$dChase(QS_{\mathcal{C}}) = dChase_9(QS_{\mathcal{C}})$$

It can be seen that $\_\colon b_1, \_\colon b_2, \_\colon b_3, \_\colon b_4, \_\colon b_5$ form a descendant chain, since $\_\colon b_i$ descendantOf $\_\colon b_{i+1}$, for each $i = 1, \ldots, 4$. Also, $originRuleId(\_\colon b_i) = originRuleId(\_\colon b_{i+1})$, for each $i = 1, \ldots, 4$. Hence, it turns out that $QS_\mathcal{C}$ is unmsafe. However, it can be seen that $originVector(\_\colon b_i) \ncong originVector(\_\colon b_j)$, for $1 \leq i \neq j \leq 5$, and hence, by definition, $QS_\mathcal{C}$ is safe with a terminating dChase. It can be noticed that during each distinct application of r_1, the vector of constants bound to the vector of variables $\langle x_{11}, \ldots, x_{14}\rangle$ are different w.r.t $\cong$. Safe quad-systems in this way are capable of recognizing such positive cases of finite dChases, which are classified as negative cases by msafe quad-systems, by also keeping track of the origin vectors of Skolem blank-nodes in its dChase.

The following property shows that for a safe quad-system, the descendance graph of any Skolem blank node in its dChase is a directed acyclic graph (DAG):

Property 4.7 (DAG property)**.** *For a safe (csafe, msafe) quad-system $QS_\mathcal{C}$, and for any blank node $b \in \mathbf{B}_{sk}(dChase(QS_\mathcal{C}))$, its descendance graph is a DAG.*

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

Since the descendance graph G of any Skolem blank node $\_\colon b \in \mathbf{B}_{sk}(dChase(QS_\mathcal{C}))$ is s.t. G is rooted at $\_\colon b$ and is acyclic, any directed path from $\_\colon b$ 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 nodes in G. The algorithm 1 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, in the resulting graph, the presence of a path from v to v'' still gives us the information that v'' is a descendant of v. 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 replaces v with k fresh nodes $v_1, ..., v_k$, and distributes the set of edges incident to v across $v_1, ..., v_k$, s.t. (i) each v_i has at-most one incoming edge (ii) all the edges incident to v are incident to some v_i, $i \in \{1, \ldots, k\}$. Whereas out going edges of v are copied for each v_i. Hence, after the above operation each v_i has an indegree 1, whereas outdegree of v_i is same as the outdegree of v, $i \in \{1, \ldots, k\}$. Hence, after all the nodes are visited, every node except the root in the new graph G has an indegree 1. G is still rooted, connected, acyclic, and is hence a tree. The algorithm terminates as there are no cycles in graph, and at some point reaches a node with no children. For instance, the unraveling of the descendance graph of $\_\colon b_4$ in Fig. 1 is shown in Fig. 2. The following property holds for any Skolem blank node of a safe quad-system.

Algorithm 1:

```
UnRavel (Descendance Graph G)
/* procedure to unravel, a descendance graph
   into a tree                                          */
Input  : descendance graph G = ⟨V, E, λ_r, λ_v, λ_c⟩
Output: A labeled Tree G
begin
    G = ⟨V, E, λ_r, λ_v, λ_c⟩ := RemoveTranstiveEdges(G);
    foreach Node v_o ∈ preOrder(G) do
        if (k = indegree(v_o)) > 1 then
            {v_1, ..., v_k} :=getFreshNodes();/* each
               v_i ∉ V is fresh                         */
            /* replace old node v_o by the fresh
               nodes in V                               */
            removeNodeFrom(v_o, V);
            addNodesTo({v_1, ..., v_k}, V);
            foreach (v_o, v') ∈ E do
                /* replace each outgoing edge
                   from v_o with a fresh outgoing
                   edges from each fresh node v_i
                   */
                removeEdgeFrom((v_o, v'), E);
                addEdgesTo({(v_1, v'), ..., (v_k, v')}, E);
            i := 1;
            foreach (v', v_o) ∈ E do
                /* replace each incoming edge of
                   v_o with an incoming edge for a
                   unique v_i                           */
                removeEdgeFrom((v', v_o), E);
                addEdgeTo((v', v_i), E);
                i++;

    /* restrict node labels to the updated set
       of nodes in V                                    */
    λ_r := λ_r|_V, λ_v := λ_v|_V, λ_c := λ_c|_V;
    return G;
```

Property 4.8. *For a safe quad-system $QS_\mathcal{C} = \langle Q_\mathcal{C}, R\rangle$, and any Skolem blank node in $dChase(QS_\mathcal{C})$, the unraveling (Algorithm 1) of its descendance graph results in a tree $t = \langle V,\, E,\, \lambda_r,\, \lambda_v,\, \lambda_c\rangle$ s.t.:*

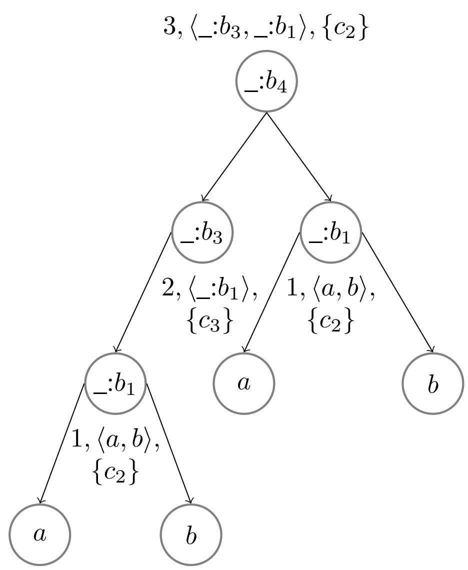

Fig. 2. descendance graph of Fig. 1 unraveled into a tree. Note: n.d. labels are not shown

1. *any leaf node of* t *is from the set* $\mathbf{C}(QS_{\mathcal{C}})$,
2. *any non-leaf node of* t *is from the set* $\mathbf{B}_{sk}(dChase(QS_{\mathcal{C}}))$,
3. $order(t) \leq w$, *where* $w = max_{r \in R}|fr(r)|$,
4. *there cannot be a path between* $b \neq b' \in V$, *with* $\lambda_r(b) = \lambda_r(b')$ *and* $\lambda_v(b) \cong \lambda_v(b')$,
5. *there cannot be a path between* $b \neq b' \in V$, *with* $\lambda_r(b) = \lambda_r(b')$, *if* $QS_{\mathcal{C}}$ *is also msafe,*
6. *there cannot be a path between* $b \neq b' \in V$, *with* $\lambda_c(b) = \lambda_c(b')$, *if* $QS_{\mathcal{C}}$ *is also csafe.*

Proof.
1. Since any node n in the descendance graph is s.t. $n \in \mathbf{C}(dChase(QS_{\mathcal{C}}))$, and since $\mathbf{C}(dChase(QS_{\mathcal{C}})) = \mathbf{C}(QS_{\mathcal{C}}) \uplus \mathbf{B}_{sk}(dChase(QS_{\mathcal{C}}))$. Since any member $m \in \mathbf{B}_{sk}(dChase(QS_{\mathcal{C}}))$ is generated from an application of a BR with an assignment μ s.t. its frontier variables are assigned by μ with a set of constants, m has at-least one child. But, since n is a leaf node, $n \in \mathbf{C}(QS_{\mathcal{C}})$.
2. Since any member $m \in \mathbf{C}(QS_{\mathcal{C}})$ cannot have descendants and since any non-leaf node has children, m cannot be a non-leaf node. Hence, non-leaf nodes should be from $\mathbf{B}_{sk}(dChase(QS_{\mathcal{C}}))$.
3. The order of t is the maximal outdegree among the nodes of t, and outdegree of a node is the number of children it has. Since any node in t with non-zero outdegree is a Skolem blank-node _: b generated by application of an assignment μ on $r = body(r)(\vec{x}, \vec{z}) \rightarrow head(r)(\vec{x}, \vec{y}) \in R$, the number of children _: b has equals $\|\vec{x}\|$. Hence, order of t is bounded by w.
4. Since any path from b to b' implies that b' is a descendant of b, it should be the case that $\lambda_r(b) \neq \lambda_r(b')$ or $\lambda_v(b) \neq \lambda_v(b')$ otherwise safety condition would be violated.
5. Similar to above, immediate by definition.
6. Similar to above, immediate by definition.

□

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

Lemma 4.9. *For any safe/msafe/csafe quad-system* $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R \rangle$, *the following holds: (i) the dChase size* $\|dChase(QS_{\mathcal{C}})\| = \mathcal{O}(2^{2^{\|QS_{\mathcal{C}}\|}})$, *(ii)* $dChase(QS_{\mathcal{C}})$ *can be computed in* 2EXPTIME, *(iii) if* $\|R\|$ *and the set of schema triples in* $Q_{\mathcal{C}}$ *is fixed to a constant, then* $\|dChase(QS_{\mathcal{C}})\|$ *is a polynomial in* $\|QS_{\mathcal{C}}\|$ *and can be computed in* PTIME.

Proof. The proofs are provided for safe quad-systems, but since CSAFE ⊂ MSAFE ⊂ SAFE and since we are giving upper bounds, they also propagate trivially to msafe and csafe quad-systems.

(i) For any blank node in $dChase(QS_{\mathcal{C}})$, the size of its originVector is upper bounded by $w = max_{r \in R}|fr(r)|$. If S is the set of all origin vectors of blank-nodes in $dChase(QS_{\mathcal{C}})$, then cardinality of the set $S' = S \backslash \cong$ is upper bounded by $(|\mathbf{U}(QS_{\mathcal{C}})| + |\mathbf{L}(QS_{\mathcal{C}})| + w)^w$, which means that $S' = \mathcal{O}(2^{\|QS_{\mathcal{C}}\|})$. Also, since the set of origin ruleId labels, $Rids$, can at most be $|R|$, hence the cardinality of the set $Rids \times S' = \mathcal{O}(2^{\|QS_{\mathcal{C}}\|})$. For the descendance tree t of any Skolem blank node of $dChase(QS_{\mathcal{C}})$, since there cannot be paths in t between distinct b and b', s.t. $originRuleId(b) = originRuleId(b')$ and $originVector(b) \cong originVector(b')$, the length of any such path is upper bounded by $Rids \times S' = \mathcal{O}(2^{\|QS_{\mathcal{C}}\|})$. However, it turns out that this above upper bound provided is loose, as there is the need of additional *filter* BRs to transform/back-propagate vectors of constants associated with Skolem blank nodes generated by repetitive application of the same BR. For instance, consider the set of BRs in eg: 4.6. The BR r_1 transforms the origin vector to a new vector each time during its application. BRs r_2 - r_5 deals with back propagation of these vectors back to input origin vectors of BR r_1. Hence, such filter BRs rule out the case of a BR being applied to a quad that contains a Skolem blank node that was generated using the same BR on an isomorphic origin vector, ensuring that the safety criteria for Skolem blank-nodes generated is not violated. It turns out that the number of such filter BRs required is polynomial w.r.t. to the number of descendants with the same rule id, for a node in t. Hence, it turns out the depth of t is polynomially bounded by $\|R\|$. (Note that depth of t is bounded by $|R|$ for msafe quad-systems. Also since, the set of origin context labels are bounded by the set of existential variables in R, depth of t is bounded by $\|R\|$ for csafe quad-systems.) Also order of the tree is bounded by w. Hence, any such tree can have at most $\mathcal{O}(2^{\|QS_\mathcal{C}\|})$ leaf nodes, $\mathcal{O}(2^{\|QS_\mathcal{C}\|})$ inner nodes, and $\mathcal{O}(2^{\|QS_\mathcal{C}\|})$ nodes. Since each of the leaf nodes can only be from $\mathbf{C}(QS_\mathcal{C})$ and each of the inner nodes correspond to an existential variable in R, the number of such possible trees are clearly bounded double exponentially in $\|QS_\mathcal{C}\|$, hence bounds the number of Skolem blank nodes generated in the dChase.

(ii) From (i) $\|dChase(QS_\mathcal{C})\|$ is double exponential in $\|QS_\mathcal{C}\|$, and since each iteration add at-least one quad to its dChase, the number of iterations are bounded double exponentially in $\|QS_\mathcal{C}\|$. Also, by lemma 3.2 any iteration i can be done in time $\mathcal{O}(\|dChase_{i-1}(QS_\mathcal{C})\|^{\|R\|})$. Hence, by using (i), we get $\|dChase_{i-1}(QS_\mathcal{C})\| = \mathcal{O}(2^{2^{\|QS_\mathcal{C}\|}})$. Hence, we can infer that each iteration i can be done in time $\mathcal{O}(2^{\|R\| * 2^{\|QS_\mathcal{C}\|}})$. Also since the number of iterations is double exponential, computing $dChase(QS_\mathcal{C})$ is in 2EXPTIME.

(iii) Since $\|R\|$ is fixed to a constant, the set of existential variables is also a constant. Also in this case, since the size of the frontier of any $r \in R$ is also a constant, the order and depth of any descendant tree t of a Skolem blank node is a constant. Hence, the number of (leaf) nodes of t is bounded by a constant. Also in this setting, the label of inner nodes of t, which correspond to existential variables, is also a constant, and the leaf nodes of t can only be a constant in $\mathbf{C}(QS_\mathcal{C})$. Hence, the number of descendant trees and consequentially, the number of Skolem blank nodes generated is bounded by $\mathcal{O}(|\mathbf{C}(QS_\mathcal{C})|^z)$, where z is a constant. Hence, the set of constants generated in $dChase(QS_\mathcal{C})$ is a polynomial in $\|QS_\mathcal{C}\|$, and so is $\|dChase(QS_\mathcal{C})\|$.

Since in any dChase iteration except the final one, at least one quad should be added, and also since the final dChase can have at most $\mathcal{O}(\|QS_\mathcal{C}\|^z)$ triples, the total number of iterations are bounded by $\mathcal{O}(\|QS_\mathcal{C}\|^z)$ (†). By lemma 3.2, since any iteration i can be computed in $\mathcal{O}(\|dChase_{i-1}(QS_\mathcal{C})\|^{\|R\|})$ time, and since $\|R\|$ is a constant, the time required for each iteration is a polynomial in $\|dChase_{i-1}(QS_\mathcal{C})\|$, which is at most a polynomial in $\|QS_\mathcal{C}\|$. Hence, any dChase iteration can be performed in polynomial time in size of $QS_\mathcal{C}$ (‡). From (†) and (‡), it can be concluded that dChase can be computed in PTIME. □

Lemma 4.10. *For any safe/msafe/csafe quad-system, the following holds: (i) data complexity of CCQ entailment is in* PTIME*, (ii) combined complexity of CCQ entailment is in* 2EXPTIME*.*

Proof. Note that the proofs are provided for safe quad-systems, but since CSAFE ⊂ MSAFE ⊂ SAFE and since we are giving upper bounds, they also propagate trivially to msafe and csafe quad-systems.

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

(i) Since for data complexity, the size of the BRs $\|R\|$, the set of schema triples, and $\|CQ()\|$ is fixed to constant. From lemma 4.9 (iii), we know that under the above mentioned settings the dChase can be computed in PTIME and is polynomial in the size of $QS_\mathcal{C}$. Since $\|CQ()\|$ is fixed to a constant, and from (†), binding the set of constants in $dChase(QS_\mathcal{C})$ on $CQ()$ still gives a number of bindings that is worst case polynomial in the size of $\|QS_\mathcal{C}\|$. Since membership of these bindings can checked in the polynomially sized dChase in PTIME, the time required for CCQ entailment is in PTIME.

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

Theorem 4.11. *For any safe/msafe/csafe quad-system, the following holds: (i) The data complexity of CCQ entailment is* PTIME*-complete (ii) The combined complexity of CCQ entailment is* 2EXPTIME*-complete.*

Proof. (i)(Membership) See lemma 4.10 for the membership in PTIME.
(Hardness) Follows from the PTIME-hardness of data complexity of CCQ entailment for Range-Restricted quad-systems (Theorem 5.2), which are contained in safe/msafe/csafe quad-systems.
(ii) (Membership) See lemma 4.10.
(Hardness) See following heading. □

4.1. 2EXPTIME-Hardness of CCQ Entailment

In this subsection, we show that the combined complexity of the decision problem of CCQ entailment for context acyclic quad-systems is 2EXPTIME-hard. We show this by reduction of the word-problem of a 2EXPTIME deterministic turing machine (DTM) to the CCQ entailment problem. A DTM M is a tuple $M = \langle Q, \Sigma, \Delta, q_0, q_A \rangle$, where

- Q is a set of states,
- Σ is a finite set of letters that includes the blank symbol $\Box$,
- $\Delta\colon (Q \times \Sigma) \rightarrow (Q \times \Sigma \times \{+1, -1\})$ is the transition function,
- $q_0 \in Q$ is the initial state.
- $q_A \in Q$ is the accepting state.

W.l.o.g. we assume that there exists exactly one accepting state, which is also the lone halting state. A configuration is a word $\vec{\alpha} \in \Sigma^* Q \Sigma^*$. A configuration $\vec{\alpha}_2$ is a successor of the configuration $\vec{\alpha}_1$, iff one of the following holds:

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

where $q, q' \in Q$, $\sigma, \sigma', \sigma_l, \sigma_r \in \Sigma$, and $\vec{w}_l, \vec{w}_r \in \Sigma^*$. Since number of configurations can at most be doubly exponential in the size of the input string, and since 2EXPTIME $\subseteq$ 2EXPSPACE, the number of tape cells traversed by the DTM tape head is also bounded double exponentially. A configuration $\vec{c} = \vec{w}_l q \vec{w}_r$ is an accepting configuration iff $q = q_A$. A language $L \subseteq \Sigma^*$ is accepted by a 2EXPTIME bounded DTM M, iff for every $\vec{w} \in L$, M accepts $\vec{w}$ in time $\mathcal{O}(2^{2^{\|\vec{w}\|}})$.

Simulating DTMs using Safe Quad-Systems Consider a double exponential time bounded DTM $M = \langle Q, \Sigma, \Delta, q_0, q_A \rangle$, and a string $\vec{w}$, with $\|\vec{w}\| = n$. In order to simulate M, we construct a quad-system $QS_{\mathcal{C}}^M = \langle Q_{\mathcal{C}}^M, R \rangle$, where $\mathcal{C} = \{c_1, ..., c_n\}$, whose various elements represents the constructs of M. We follow the technique in works such as [34,36] to iteratively generate a doubly exponential number of objects that represent the cells of the tape of the DTM. Let $Q_{\mathcal{C}}^M$ be initialized with the following quads:

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

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

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

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

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

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

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

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

Each of the above set rules are instantiated for $0 \leq i < n$, and in this way after n generating dChase iterations, c_n has doubly exponential number of elements of type R, that are ordered linearly using the relation $succ_n$. By virtue of the first rule below, each of the objects representing the cells of the DTM are linearly ordered by the relation $succ$. Also the transitive closure of $succ$ is defined as the relation relation $succt$

$$c_n\colon (x_0, succ_n, x_1) \to c_n\colon (x_0, succ, x_1)$$
$$c_n\colon (x_0, succ, x_1) \to c_n\colon (x_0, succt, x_1)$$
$$c_n\colon (x_0, succt, x_1), c_n\colon (x_1, succt, x_2)$$
$$\to c_n\colon (x_0, succt, x_2)$$

Also using a similar construction, we could reuse the $2^{2^{n-1}}$ linearly ordered elements in c_{n-1} to create another linearly ordered chain of double exponential number of objects in c_n that represents configurations of M, whose minimal element is of type $conInit$, and the linear order relation being $conSucc$.

Various triple patterns that are used to encode the possible configurations, runs and their relations in M are:

- $(x_0, head, x_1)$ denotes the fact that in configuration x_0, the head of the DTM is at cell x_1.
- $(x_0, state, x_1)$ denotes the fact that in configuration x_0, the DTM is in state x_1.
- (x_0, σ, x_1) where $\sigma \in \Sigma$, denotes the fact that in configuration x_0, the cell x_1 contains σ.
- $(x_0, succ, x_1)$ denotes the linear order between cells of the tape.
- $(x_0, succt, x_1)$ denotes the transitive closure of $succ$.
- $(x_0, conSucc, x_1)$ to denote the fact that x_1 is a successor configuration of x_0.
- $(x_0, \texttt{rdf:type}, Accept)$ denotes the fact that the configuration x_0 is an accepting configuration.

Since in our construction, each $\sigma \in \Sigma$ is represented as relation, we could constrain that no two letters $\sigma \neq \sigma'$ are on the same cell using the following axiom:

$$c_n\colon (z_1, \sigma, z_2), c_n\colon (z_1, \sigma', z_2) \to$$

for each $\sigma \neq \sigma' \in \Sigma$. Note that the above BR has an empty head, is equivalent to asserting the negation of its body.

Initialization Suppose the initial configuration is $q_0\vec{w}\Box$, where $\vec{w} = \sigma_0...\sigma_{n-1}$, then we enforce this using the following BRs in our quad-system QS_C^M as:

$$c_n\colon (x_0, \texttt{rdf:type}, conInit), c_n\colon (x_1, \texttt{rdf:type}, min_n) \to c_n\colon (x_0, head, x_1), c_n\colon (x_0, state, q_0)$$

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

$$c_n\colon (x_j, \texttt{rdf:type}, conInit), c_n\colon (x_j, \Box, x_0), c_n\colon (x_0, succt, x_1) \to c_n\colon (x_j, \Box, x_1)$$

The last BR copies the $\Box$ to every succeeding cell in the initial configuration.

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

$$c_n\colon (x_0, head, x_i), c_n\colon (x_0, \sigma, x_i), c_n\colon (x_0, state, q), c_n\colon (x_j, succ, x_i), c_n\colon (x_0, conSucc, x_1) \to c_n\colon (x_1, head, x_j), c_n\colon (x_1, \sigma', x_i), c_n\colon (x_1, state, q_j)$$

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

$$c_n\colon (x_0, head, x_i), c_n\colon (x_0, \sigma, x_i), c_n\colon (x_0, state, q), c_n\colon (x_i, succ, x_j), c_n\colon (x_0, conSucc, x_1), \to c_n\colon (x_1, head, x_j), c_n\colon (x_1, \sigma', x_i), c_n\colon (x_1, state, q_j)$$

Inertia If in any configuration the head is at cell i of the tape, then in every successor configuration, elements in preceding and following cells of i in the tape are retained. The following two BRs ensures this:

$$c_n\colon (x_0, head, x_i), c_n\colon (x_0, conSucc, x_1), c_n\colon (x_j, succt, x_i), c_n\colon (x_0, \sigma, x_j) \to c_n\colon (x_1, \sigma, x_j)$$
$$c_n\colon (x_0, head, x_i), c_n\colon (x_0, conSucc, x_1), c_n\colon (x_i, succt, x_j), c_n\colon (x_0, \sigma, x_j) \to c_n\colon (x_1, \sigma, x_j)$$

The rules above are instantiated for every $\sigma \in \Sigma$.

Acceptance A configuration whose state is q_A is accepting:

$$c_n\colon (x_0, state, q_A) \to c_n\colon (x_0, \texttt{rdf:type}, Accept)$$

If a configuration of accepting type is reached, then it can be back propagated to the initial configuration, using the following BR:

$$c_n\colon (x_0, conSucc, x_1), c_n\colon (x_1, \texttt{rdf:type}, Accept)$$
$$\rightarrow c_n\colon (x_0, \texttt{rdf:type}, Accept)$$

Finally since M accepts $\vec{w}$ iff the initial configuration is an accepting configuration. Let CQ^M be CCQ: $\exists y\ c_n(y, \texttt{rdf:type}, conInit), c_n\colon (y, \texttt{rdf:type}, Accept)$. It can easily be verified that $QS_{\mathcal{C}}^M \models CQ^M$ iff the initial configuration is an accepting configuration. In order to prove the soundness and completeness of our simulation, we prove the following claims:

Claim (1) The quad-system $QS_{\mathcal{C}}^M$ in the aforementioned simulation is a csafe quad-system

It can be noted that only BRs in which existentials are present are the BRs used to generate the double exponential chain of tape cells and configurations, and are of the form (eBr). Note that in each of application of such a BR, a blank-node $\_\colon b$ generated in a context c_i, for any $i = 1, \ldots, n$, is s.t. $originContexts(\_\colon b) = \{c_i\}$ and has exactly two child blank-nodes, each of whose origin contexts is $\{c_{i-1}\}$. Hence, any Skolem blank-node generated in any c_i, for $i = 1 \ldots n$ is s.t. its child blank-nodes has origin contexts c_{i-1}. Thanks to the above property, it turns out there exists no two blank-nodes $\_\colon b, \_\colon b'$ in the dChase of $QS_{\mathcal{C}}^M$ s.t. $\_\colon b$ is a descendant of $\_\colon b'$ and $originContexts(\_\colon b) = originContexts(\_\colon b')$. Therefore $QS_{\mathcal{C}}^M$ is csafe.

Claim (2) $QS_{\mathcal{C}}^M \models CQ^M$ iff M accepts $\vec{w}$.

Suppose that $QS_{\mathcal{C}}^M \models CQ^M$, then it holds that in any model $\mathcal{I}^{\mathcal{C}} = \{I^{c_i}\}_{i=1\ldots n}$ of $QS_{\mathcal{C}}^M$, $\mathcal{I}^{\mathcal{C}} \models CQ^M$, which implies that I^{c_n} has an object o in its domain s.t. $o \in conInit^{c_n}$ and $o \in Accept^{c_n}$. But thanks to the acceptance axioms it follows that there exists an object o' s.t. (o, o') in the reflexive-transitive closure of $conSucc^{c_n}$ s.t. $o' \in Accept^{c_n}$. Also thanks to the initialization axioms, it can be seen that o represents the initial configuration of M i.e. it represents the configuration in which the initial state is q_0, and the left end of the read-write tape contains $\vec{w}$ followed by trailing $\Box$s, with the read-write head positioned at the first cell of the tape. Also the transition axioms makes sure that if $(o, o'') \in conSucc^{c_n}$, then o'' represents a successor configuration of o. That is, if o represents the configuration in which M is at state q with read-write head at position pos of the tape that contains a letter $\sigma \in \Sigma$, and if $\Delta(q, \sigma) = (q', \sigma', D)$, then o'' represents the configuration in which M is at state q', which read-write head at the position $pos - 1/pos + 1$ depending on whether $D = -1/+1$, and σ' at the position $pos - 1/pos + 1$ of the tape. As a consequence of the above arguments, it follows that o' represents an accepting configuration of M, i.e. a configuration in which the state is q_A, the lone accepting, halting state. This means that M accepts the string $\vec{w}$.

For the converse, we briefly show that if $QS_{\mathcal{C}}^M \not\models CQ^M$ then M does not accept $\vec{w}$. Suppose that $QS_{\mathcal{C}}^M \not\models CQ^M$, then this implies that there exists a model $\mathcal{I}^{\mathcal{C}} = \{I^{c_i}\}_{i=1\ldots n}$ of $QS_{\mathcal{C}}^M$, s.t. $\mathcal{I}^{\mathcal{C}} \not\models CQ^M$. This means that no object in the domain of I^{c_n} exists that is a member of both $conInit^{c_n}$ and $Accept^{c_n}$. By the initialization axioms, we know that there exists an object o in the domain of I^{c_n} with $o \in conInit^{c_n}$ and by preceding discussion, we know that o represents the initial configuration of M. Also by the initial construction axioms of $QS_{\mathcal{C}}^M$, we know that o is the initial element of a double exponential chain of objects that are linearly ordered by property symbol $conSucc$. From transition axioms we know that for any o'' s.t. $(o, o'') \in conSucc^{c_n}$, then o'' represents a valid successor configuration of o, which itself holds for o'', and so on. This means that for none of the succeeding double exponential configurations of M, the accepting state q_A holds. This means that M does not reach an accepting configuration with string $\vec{w}$, and hence rejects it.

Since the construction above shows the existence of a polynomial time reduction of the word problem of a 2EXPTIME DTM, which is a 2EXPTIME-hard problem, to the CCQ entailment problem over csafe quad-systems, it immediately follows that CCQ entailment over csafe/msafe/safe quad-systems is 2EXPTIME-hard.

4.2. Procedure for detecting safe/msafe/csafe quad-systems

In this subsection, we present a procedure for deciding whether a given quad-system is safe (resp. msafe, resp. csafe) or not. If the quad-system is safe (resp. msafe, resp. csafe), the result of the procedure is a *safe dChase* (resp. *msafe dChase*, *csafe dChase*) that contains the standard dChase, and can be used for query answering. Since safety (resp. msafety, resp. csafety) 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 ruleIds and origin vectors (resp. origin ruleIds, resp. origin-contexts) of each Skolem blank node generated. In each iteration, a test for safety is performed, by checking the presence of Skolem blank-nodes that violates the safety (resp. msafety, resp. csafety) condition. In case a violation is detected, a distinguished constant is generated and the safe (resp. msafe, resp. csafe) dChase construction is aborted, prematurely. On the contrary, if there exists an iteration in which no new quad is generated, the safe (resp. msafe, resp. csafe) dChase computation stops with a completed safe (resp. msafe, resp. csafe) dChase that contains the standard dChase. Since all the additional quads produced for accounting information, uses a distinguished context identifier $c_c \notin \mathcal{C}$, the computed safe (resp. msafe, resp. csafe) dChase itself can be used for standard query answering. Before geting to the details of the procedure, we give a few necessary definitions.

Definition 4.12 (Context Scope)**.** *The context scope of a term t in a set of quad-patterns Q, denoted by $cScope(t, Q)$ is given as: $cScope(t, Q) = \{c \mid c\colon (s, p, o) \in Q, s = t \vee p = t \vee o = t\}$.*

For any quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$, let c_c be an ad hoc context identifier s.t. $c_c \notin \mathcal{C}$, then for $r_i = body(r_i)(\vec{x}, \vec{z}) \rightarrow head(r_i)(\vec{x}, \vec{y}) \in R$, we define transformations $augS(r_i)$, $augM(r_i)$, $augC(r_i)$ as follows:

$$augS(r_i) = body(r_i)(\vec{x}, \vec{z}) \rightarrow head(r_i)(\vec{x}, \vec{y}) \wedge \forall y_j \in \{\vec{y}\}\ [\bigwedge_{x_k \in \{\vec{x}\}} c_c\colon (x_k, \text{descendantOf}, y_j)\ \wedge\ c_c\colon (y_j, \text{descendantOf}, y_j)\ \wedge c_c\colon (y_j, \text{originRuleId}, i)\ \wedge\ c_c\colon (y_j, \text{originVector}, \vec{x})]$$

It should be noted that $c_c\colon (y_j,$ originVector, $\vec{x})$ is not a valid quad pattern, and is only used for notation brevity. In the actual implementation, vectors can be stored using an rdf container data structure such as `rdf:List`, `rdf:Seq` or by typecasting it as a string.

$$augM(r_i) = body(r_i)(\vec{x}, \vec{z}) \rightarrow head(r_i)(\vec{x}, \vec{y}) \wedge \forall y_j \in \{\vec{y}\}\ [\bigwedge_{x_k \in \{\vec{x}\}} c_c\colon (x_k, \text{descendantOf}, y_j)\ \wedge\ c_c\colon (y_j, \text{descendantOf}, y_j)\ \wedge c_c\colon (y_j, \text{originRuleId}, i)]$$

$$augC(r_i) = body(r_i)(\vec{x}, \vec{z}) \rightarrow head(r_i)(\vec{x}, \vec{y}) \wedge \forall y_j \in \{\vec{y}\}, \forall c \in cScope(y_j, head(r_i))\ [\bigwedge_{x_k \in \{\vec{x}\}} c_c\colon (x_k, \text{descendantOf}, y_j) \wedge\ c_c\colon (y_j, \text{descendantOf}, y_j)\ \wedge\ c_c\colon (y_j, \text{originContext}, c)]$$

Intuitively, the transformation $augS/augM/augC$ on a BR r_i, augments the head part of r_i with additional types of quad patterns, which are the following:

1. $c_c\colon (x_k, \text{descendantOf}, y_j)$, for every existentially quantified variable y_j in $\vec{y}$ and universally quantified variable $x_k \in \{\vec{x}\}$. This is done because, during dChase computation any application of an assignment μ on r_i s.t. $\vec{x}[\mu] = \vec{a}$, resulting in the generation of a Skolem blank node $\_\colon b = \mu^{ext(\vec{y})}(y_j)$, any $a_i \in \{\vec{a}\}$ is a descendant of $\_\colon b$. Hence, due to these additional quad-patterns, quads of the form $c_c\colon (a_i$, descendantOf, $\_\colon b)$ are also produced, and in this way, keeps track of the descendants of any Skolem blank node produced.
2. $c_c\colon (y_j$, descendantOf, $y_j)$, in order to maintain also the reflexivity of 'descendantOf' relation.
3. $c_c\colon (y_j, \text{originContext}, c)$, for every existentially quantified variable y_j in $\{\vec{y}\}$, every $c \in cScope(y_j,\ head(r_i))$. This is done because during dChase computation, any application of an assignment μ on r_i, s.t. $\vec{x}[\mu] = \vec{a}$, resulting in the generation of a Skolem blank node $\_\colon b = \mu^{ext(\vec{y})}(y_j)$, c is an origin context of $\_\colon b$, Hence, due to these additional quad-patterns, quads of the form $c_c\colon (\_\colon b, \text{originContext}, c)$ is also produced. In this way, keeps track of the origin-contexts of any Skolem blank node produced.
4. $c_c\colon (y_j$, originVector, $\vec{x})$, This is done because during dChase computation, any application of an assignment μ on r_i, s.t. $\vec{x}[\mu] = \vec{a}$, resulting in the generation of a Skolem blank node $\_\colon b = \mu^{ext(\vec{y})}(y_j)$, $\vec{a}$ is the origin vector of $\_\colon b$. Hence, due to these additional quad-patterns, quads of the form $c_c\colon (\_\colon b$, originVector, $\vec{a})$ is also produced. In this way, keeps track of the origin vector of any Skolem blank node produced.
5. $c_c\colon (y_j$, originRuleId, $i)$, for every existentially quantified variable y_j in $\{\vec{y}\}$, inorder to keep track of the ruleId of the BR used to create any Skolem blank node.

It can be noticed that for any BR r_i without existentially quantified variables, the transformations $augS/augM/augC$ leaves r_i unchanged. For any set of BRs R, let

$$augS(R) \text{ (resp. } augM(R), \text{resp. } augC(R)) = \bigcup_{r_i \in R} augS(r_i) \text{ (resp. } augM(r_i), \text{resp. } augC(r_i)) \cup \{c_c\colon (x_1, \text{descendantOf}, z_1) \wedge c_c\colon (z_1, \text{descendantOf}, x_2) \rightarrow c_c\colon (x_1, \text{descendantOf}, x_2)\}$$

The function unSafeTest (resp. unMSafeTest, resp. unCSafeTest) defined below, given a BR $r_i = body(r_i)(\vec{x}, \vec{z}) \rightarrow head(r_i)(\vec{x}, \vec{y})$, an assignment μ, and a quad-graph Q checks, if application of μ on r_i violates the safety (resp. msafety, resp. csafety) condition on Q.
unSafeTest(r_i, μ, Q)=True iff $\exists \_\colon b, \_\colon b' \in \mathbf{B}$, with all the following conditions being satisfied:

- $\_\colon b \in \{\vec{x}[\mu]\}$, and
- $c_c\colon (\_\colon b', \text{descendantOf}, \_\colon b) \in Q$, and
- $c_c\colon (\_\colon b', \text{originRuleId}, i) \in Q$, and
- $c_c\colon (\_\colon b', \text{originVector}, \vec{a}) \in Q$, and $\vec{a} \cong \vec{x}[\mu]$.

Intuitively, unSafeTest returns True, if μ applied to r will produce a fresh Skolem blank node $\_\colon b''$, whose child $\_\colon b \in \vec{x}[\mu]$, and according to knowledge in Q, $\_\colon b'$ a descendant of $\_\colon b$ s.t. origin ruleId of $\_\colon b'$ is i (which is also the origin ruleId of $\_\colon b''$) and origin vector of $\_\colon b'$ is isomorphic to origin vector of $\vec{x}[\mu]$ (which is also the origin vector of $\_\colon b''$). The functions unMSafeTest and unCSafeTest are similarly defined as follows:
unMSafeTest(r_i, μ, Q)=True iff $\exists \_\colon b, \_\colon b' \in \mathbf{B}$, with all the following conditions being satisfied:

- $\_\colon b \in \{\vec{x}[\mu]\}$, and
- $c_c\colon (\_\colon b', \text{descendantOf}, \_\colon b) \in Q$, and
- $c_c\colon (\_\colon b', \text{originRuleId}, i) \in Q$.

unCSafeTest(r_i, μ, Q)=True iff $\exists \_\colon b, \_\colon b' \in \mathbf{B}$, $\exists y_j \in \{\vec{y}\}$, with all the following being satisfied:

- $\_\colon b \in \{\vec{x}[\mu]\}$, and
- $c_c\colon (\_\colon b', \text{descendantOf}, \_\colon b) \in Q$, and
- $\{c \mid c_c\colon (\_\colon b', \text{originContext}, c) \in Q\} = cScope(\, y_j, head(r_i)(\vec{x}, \vec{y})) \setminus \{c_c\}$.

For any BR r_i and an assignment μ, the *safe/msafe/csafe application* of μ on r_i w.r.t. a quad-graph $Q_{\mathcal{C}}$ is defined as follows:

$$apply^{\text{safe}}(r_i, \mu, Q_{\mathcal{C}}) = \begin{cases} \textbf{unSafe}, & \text{If unSafeTest}(r_i, \mu, Q_{\mathcal{C}}) = \text{True;} \\ apply(r_i, \mu), & \text{Otherwise;} \end{cases}$$

$$apply^{\text{msafe}}(r_i, \mu, Q_{\mathcal{C}}) = \begin{cases} \textbf{unMSafe}, & \text{If unMSafeTest}(r_i, \mu, Q_{\mathcal{C}}) = \text{True;} \\ apply(r_i, \mu), & \text{Otherwise;} \end{cases}$$

$$apply^{\text{csafe}}(r_i, \mu, Q_{\mathcal{C}}) = \begin{cases} \textbf{unCSafe}, & \text{If unCSafeTest}(r_i, \mu, Q_{\mathcal{C}}) = \text{True;} \\ apply(r_i, \mu), & \text{Otherwise;} \end{cases}$$

where **unSafe** $= c_c\colon (\text{unsafe}, \text{unsafe}, \text{unsafe})$ (resp. **unMSafe** $= c_c\colon (\text{unmsafe}, \text{unmsafe}, \text{unmsafe})$, resp. **unCSafe** $= c_c\colon (\text{uncsafe}, \text{uncsafe}, \text{uncsafe})$) is a distinguished quad that is generated, if the prerequisites of safety (resp. msafety, resp. csafety) is violated. For any quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R \rangle$, we define its *safe dChase* $dChase^{\text{safe}}(QS_{\mathcal{C}})$ as follows:
$dChase_0^{\text{safe}}(QS_{\mathcal{C}}) = Q_{\mathcal{C}}$;
$dChase_{m+1}^{\text{safe}}(QS_{\mathcal{C}}) = dChase_m^{\text{safe}}(QS_{\mathcal{C}}) \cup apply^{\text{safe}}(r_i, \mu, dChase_m^{\text{safe}}(QS_{\mathcal{C}}))$, if there exists $r_i \in augS(R)$, assignment μ s.t. $applicable_{augS(R)}(r_i, \mu, dChase_m^{\text{safe}}(QS_{\mathcal{C}}))$;
$dChase_{m+1}^{\text{safe}}(QS_{\mathcal{C}}) = dChase_m^{\text{safe}}(QS_{\mathcal{C}})$, otherwise; for any $m \in \mathbb{N}$.
$dChase^{\text{safe}}(QS_{\mathcal{C}}) = \bigcup_{m \in \mathbb{N}} dChase_m^{\text{safe}}(QS_{\mathcal{C}})$
The termination condition for safe dChase computation can be implemented using the following conditional: If there exists m s.t.
$dChase_m^{\text{safe}}(QS_{\mathcal{C}}) = dChase_{m+1}^{\text{safe}}(QS_{\mathcal{C}})$; then
$dChase^{\text{safe}}(QS_{\mathcal{C}}) = dChase_m^{\text{safe}}(QS_{\mathcal{C}})$.
Similarly, dChases $dChase^{\text{msafe}}(QS_{\mathcal{C}})$ and $dChase^{\text{csafe}}(QS_{\mathcal{C}})$ are defined for msafe and csafe quad-systems, respectively.
The following theorem shows that the procedure above described for detecting unsafe quad-systems is sound and complete:

Theorem 4.13. *For any quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R \rangle$, the quad **unSafe** (resp. **unMSafe**, resp. **unCSafe**) $\in dChase^{safe}(QS_{\mathcal{C}})$ (resp. $dChase^{msafe}(QS_{\mathcal{C}})$, resp. $dChase^{csafe}(QS_{\mathcal{C}})$), iff $QS_{\mathcal{C}}$ is unsafe (resp. unmsafe, resp. uncsafe).*

It should be noted that for any quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R \rangle$, $dChase^{\text{safe}}(QS_{\mathcal{C}})$ (resp. $dChase^{\text{msafe}}(QS_{\mathcal{C}})$, resp. $dChase^{\text{csafe}}(QS_{\mathcal{C}})$) is a finite set and hence the iterative procedure which we described earlier terminates, regardless of whether $QS_{\mathcal{C}}$ is safe (resp. msafe, resp. csafe) or not. This is because if $QS_{\mathcal{C}}$ is safe (resp. msafe, resp. csafe), then, as we have seen before, there exists a double exponential bound on number of quads in its dChase. Hence, there is an iteration in which no new quad is generated, which leads to stopping of computation. Otherwise, if $QS_\mathcal{C}$ is unsafe (resp. msafe, resp. csafe), then from theorem 4.13, we know that the quad **unSafe** (resp. **unMSafe**, resp. **unCSafe**) gets generated in $dChase^{\text{safe}}(QS_\mathcal{C})$ (resp. $dChase^{\text{msafe}}(QS_\mathcal{C})$, resp. $dChase^{\text{csafe}}(QS_\mathcal{C})$). This implies that there exists an iteration m s.t. the quad **unSafe** (resp. **unMSafe**, resp. **unCSafe**) is in $dChase^{\text{safe}}_m(QS_\mathcal{C})$ (resp. $dChase^{\text{msafe}}_m(QS_\mathcal{C})$, resp. $dChase^{\text{csafe}}_m(QS_\mathcal{C})$). W.l.o.g, let m be the first such iteration. This means that there exists a BR $r_i \in R$ with head $head(r_i)(\vec{x}, \vec{y})$, assignment μ s.t. $applicable_{augS(R)}(r_i, \mu, dChase^{\text{safe}}_{m-1}(QS_\mathcal{C}))$ (resp. $applicable_{augM(R)}(r_i, \mu, dChase^{\text{msafe}}_{m-1}(QS_\mathcal{C}))$, resp. $applicable_{augC(R)}(r_i, \mu, dChase^{\text{csafe}}_{m-1}(QS_\mathcal{C}))$ holds. By construction, since $head(r_i)[\mu^{ext(\vec{y})}]$ is not generated, and instead the quad **unSafe** (resp. **unMSafe**, resp. **unCSafe**) is generated, $applicable_{augS(R)}(r_i, \mu, dChase^{\text{safe}}_m(QS_\mathcal{C}))$ (resp. $applicable_{augM(R)}(r_i, \mu, dChase^{\text{msafe}}_m(QS_\mathcal{C}))$, resp. $applicable_{augC(R)}(r_i, \mu, dChase^{\text{csafe}}_m(QS_\mathcal{C}))$ holds yet again. This means that the termination condition is satisfied at iteration $m+1$, and hence computation stops. Note that regardless of whether a given quad-system is safe (resp. msafe, resp. csafe) or not, the number of safe (resp. msafe, resp. csafe) dChase iterations is double exponentially bounded in the size of the quad-system.

Hence, after running procedure described above, if the quad **unSafe** (resp. **unMSafe**, resp. **unCSafe**) is not generated, then its safe (resp. msafe, resp. csafe) dChase itself can be used for CCQ answering, as in such a case the standard dChase is contained in safe (resp. msafe, resp. csafe) dChase, and all the quads generated for accounting information has the context identifier c_c. Hence, for any safe (resp. msafe, resp. csafe) quad-system, for any boolean CCQ that does not contain quad patterns of the form $c_c\colon (s, p, o)$, dChase entails CCQ iff safe (resp. msafe, resp. csafe) dChase entails CCQ.

5. Range Restricted Quad-Systems: Restricting to Range Restricted BRs

In this section, we investigate the complexity of CCQ entailment over quad-systems, whose BRs do not have existentially quantified variables. Such BRs are of the form:

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

Note that any set of BRs R of the form above can be replaced by semantically equivalent set R', s.t. each $r \in R'$ is the form:

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

Also $\|R'\|$ is at most quadratic in $\|R\|$, and hence, w.l.o.g, we assume that each $r \in R$ is of the form (4). Borrowing the parlance from the $\forall\exists$ rules setting, where rules whose variables in the head part are contained in the variables in the body part are called range restricted rules [14], we call such BRs *range restricted* (RR) BRs. We call a quad-system whose BRs are all of RR-type, a *RR quad-system*. Since there exists no existentially quantified variables in BRs of a RR quad-system, no Skolem blank nodes are produced during dChase computation. Hence, there can be no violation of the safety/msafety/csafety condition in section 4, and hence, the class of RR quad-systems are contained in the class of safe/msafe/csafe quad-systems, and is also a FEC. Of course, this containment is strict as any quad-system that contains a BR with an existential variable is not RR. We in the following see that restricting to RR BRs, size of the dChase becomes polynomial w.r.t. size of the input quad-system, and the complexity of CCQ entailment further reduces compared to safe/msafe/csafe quad-systems.

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

Proof. (i) Note that the number of constants in $QS_\mathcal{C}$ is roughly equal to $\|QS_\mathcal{C}\|$. As no existential variable occur in any BR in a RR quad-system $QS_\mathcal{C}$, the set of constants $\mathbf{C}(dChase(QS_\mathcal{C}))$ is contained in $\mathbf{C}(QS_\mathcal{C})$. Since each $c\colon (s, p, o) \in dChase(QS_\mathcal{C})$ is s.t. $c, s, p, o \in \mathbf{C}(QS_\mathcal{C})$, $|dChase(QS_\mathcal{C})| = \mathcal{O}(|\mathbf{C}(QS_\mathcal{C})|^4)$. Hence, $\|dChase(QS_\mathcal{C})\| = \mathcal{O}(|\mathbf{C}(QS_\mathcal{C})|^4) = \mathcal{O}(\|QS_\mathcal{C}\|^4)$.

(ii) Since from (i) $|dChase(QS_\mathcal{C})| = \mathcal{O}(\|QS_\mathcal{C}\|^4)$, and in each iteration of the dChase at least one new quad should be added, the number of iterations cannot exceed $\mathcal{O}(\|QS_\mathcal{C}\|^4)$. Since by lemma 3.2, each iteration i of dChase computation requires $\mathcal{O}(|R| * \|dChase_{i-1}(QS_\mathcal{C})\|^{rs})$ time, where $rs = max_{r\in R}\|r\|$, and $rs \leq \|QS_\mathcal{C}\|$, time required for each iteration is of the order $\mathcal{O}(2^{\|QS_\mathcal{C}\|})$ time. Although the number of iterations is a polynomial, each iteration requires an exponential amount of time w.r.t $\|QS_\mathcal{C}\|$.

Hence, time complexity of dChase computation is in EXPTIME.

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

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

Proof. (Membership) Follows from the membership in P of data complexity of CCQ entailment for safe quad-systems, whose expressivity subsumes the expressivity of RR quad-systems (Theorem 4.11).

(Hardness) In order to prove P-hardness, we reduce a well known P-complete problem, 3HornSat, i.e. the satisfiability of propositional Horn formulas with at most 3 literals. Note that a (propositional) Horn formula is a propositional formula of the form:

$$P_1 \wedge \ldots \wedge P_n \rightarrow P_{n+1} \tag{5}$$

where P_i, for $1 \leq i \leq n+1$, are either propositional variables or constants t, f, that represents true and false, respectively. Note that for any propositional variable P, the fact that "P holds" is represented by the formula $t \rightarrow P$, and "P does not hold" is represented by the formula $P \rightarrow f$. A 3Horn formula is a formula of the form (5), where $1 \leq n \leq 2$. Note that any (set of) Horn formula(s) Φ can be transformed in polynomial time to a polynomially sized set Φ' of 3Horn formulas, by introducing auxiliary propositional variables s.t. Φ is satisfiable iff Φ' is satisfiable. A pure 3Horn formula is a 3Horn formula of the form 5, where $n = 2$. Any 3Horn formula ϕ that is not pure can be trivially converted to equivalent pure form by appending a $\wedge\, t$ on the body part of ϕ. For instance, $P \rightarrow Q$, can be converted to $P \wedge t \rightarrow Q$. Hence, w.l.o.g. we assume that any set of 3Horn formulas is pure, and is of the form:

$$P_1 \wedge P_2 \rightarrow P_3 \tag{6}$$

We, in the following, reduce the satisfiability problem of pure 3Horn formulas to CCQ entailment problem over a quad-system whose set of schema triples, the set of BRs, and the CCQ CQ are all fixed.

For any set of pure Horn formulas Φ, we construct the quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$, where $\mathcal{C} = \{c_t, c_f\}$. For any formula $\phi \in \Phi$ of the form (6), $Q_{\mathcal{C}}$ contains a quad $c_f\colon (P_1, P_2, P_3)$. In addition $Q_{\mathcal{C}}$ contains a quad $c_t\colon (t, \texttt{rdf:type}, T)$. R is the singleton that contains only the following fixed BR:

$$c_t\colon (x_1, \texttt{rdf:type}, T), c_t\colon (x_2, \texttt{rdf:type}, T),$$
$$c_f\colon (x_1, x_2, x_3) \rightarrow c_t\colon (x_3, \texttt{rdf:type}, T)$$

Let the CQ be the fixed query $c_t : (f, \texttt{rdf:type}, T)$.

Now, it is easy to see that $QS_{\mathcal{C}} \models CQ$, iff Φ is not satisfiable. □

Theorem 5.3. *Combined complexity of CCQ entailment over RR quad-systems is* EXPTIME*-complete.*

Proof. (Membership) By lemma 5.1, for any RR quad-system $QS_{\mathcal{C}}$, its dChase $dChase(QS_{\mathcal{C}})$ can be computed in EXPTIME. Also by lemma 5.1, its dChase size $\|dChase(QS_{\mathcal{C}})\|$ is a polynomial w.r.t to $\|QS_{\mathcal{C}}\|$. Since a boolean CCQ $CQ()$ can naively be evaluated by grounding the set of constants in the dChase to the variables in the $CQ()$, and then checking if any of these groundings are contained in $dChase(QS_{\mathcal{C}})$. The number of such groundings can at most be $\|dChase(QS_{\mathcal{C}})\|^{\|CQ()\|}$ (†). Since $\|dChase(QS_{\mathcal{C}})\|$ is a polynomial in $\|QS_{\mathcal{C}}\|$, there are an exponential number of groundings w.r.t $\|CQ()\|$. Since containment of each of these groundings can be checked in time polynomial w.r.t. the size of $dChase(QS_{\mathcal{C}})$, and since $\|dChase(QS_{\mathcal{C}})\|$ is a polynomial w.r.t. $\|QS_{\mathcal{C}}\|$, 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 a DTM. The proof can slightly be modified to simulate an EXPTIME DTM. The steps in the proof is same as the one in Dantsin et al. [25], where EXPTIME-hardness of function-free Horn logic programs (Datalog) are shown. □

5.1. Restricted RR Quad-Systems

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

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

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

(Membership) for any $QS_\mathcal{C}$ whose rules are of restricted RR-type, size of any $r \in R$ is a constant. Hence, by lemma 3.2, any dChase iteration can be computed in PTIME. Since, number of iterations are also polynomial in $\|QS_\mathcal{C}\|$, $dChase(QS_\mathcal{C})$ can be computed in PTIME in the size of $QS_\mathcal{C}$ and $dChase(QS_\mathcal{C})$ has a polynomial number of constants. Hence, if we guess an assignment μ for all the existential variables in CCQ $CQ()$, to the set of constants in $dChase(QS_\mathcal{C})$. Then, one can evaluate the CCQ, by checking if $c\colon (s,p,o) \in dChase(QS_\mathcal{C})$, for each $c\colon (s,p,o) \in CQ()[\mu]$, which can be done in time $\mathcal{O}(\|CQ\| * \|dChase(QS_\mathcal{C})\|)$, and is hence is in non-deterministic PTIME, which implies that DP' is in NP.

(Hardness) We show that DP' is NP-hard, by reducing the well known NP-hard problem, 3-colorability to DP'. Given a graph $G = \langle V, E\rangle$, where $V = \{v_1, ..., v_n\}$ is the set of nodes, $E \subseteq V \times V$ is the set of edges, 3-colorability problem, is to decide if there exists a labeling function $l : V \rightarrow \{r, b, g\}$ that assigns each $v \in V$ to an element in $\{r, b, g\}$ s.t. the condition: $(v, v') \in E \rightarrow l(v) \neq l(v')$, for each $(v, v') \in E$, is satisfied.

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

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

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

6. Quad-Systems and Forall-Existential rules: A formal comparison

In this section, we formally compare the formalism of quad-systems with forall-existential (∀∃) rules, which are also called Tuple generating dependencies (Tgds)/Datalog+- rules. ∀∃ rules is a fragment of first order logic in which every formula is restricted to a certain syntactic form. A ∀∃ rule is a first order formula of the form:

$$\forall\vec{x}\forall\vec{z}\,[p_1(\vec{x},\vec{z}) \wedge ... \wedge p_n(\vec{x},\vec{y}) \rightarrow \exists\vec{y}\,p'_1(\vec{x},\vec{y}) \wedge ... \wedge p'_m(\vec{x},\vec{y})] \quad (7)$$

where $\vec{x}, \vec{y}, \vec{z}$ are vectors of variables s.t. $\{\vec{x}\}$, $\{\vec{y}\}$ and $\{\vec{z}\}$ are pairwise disjoint, $p_i(\vec{x},\vec{z})$, for $1 \leq i \leq n$ are predicate atoms whose variables are from $\vec{x}$ or $\vec{z}$, $p'_1(\vec{x},\vec{y})$, for $1 \leq i \leq m$ are predicate atoms whose variables are from $\vec{x}$ or $\vec{y}$. We, for short, occasionally note a ∀∃ rule of the form (7) as $\phi(\vec{x}, \vec{z}) \rightarrow \psi(\vec{x}, \vec{y})$, where $\phi(\vec{x}, \vec{z}) = \{p_1(\vec{x},\vec{z}), ..., p_n(\vec{x},\vec{y})\}$, $\psi(\vec{x}, \vec{y}) = \{p'_1(\vec{x},\vec{y}), ...\ p'_m(\vec{x},\vec{y})\}$. A set of ∀∃ rules is called a ∀∃ rule set. In the realm of ∀∃ rule sets, a conjunctive query (CQ) is an expression of the form:

$$\exists\vec{y}\,p_1(\vec{x},\vec{y}) \wedge ... \wedge p_r(\vec{x},\vec{y}) \quad (8)$$

where $p_i(\vec{x},\vec{y})$, for $1 \leq i \leq r$ are predicate atoms over vectors $\vec{x}$ or $\vec{y}$. A boolean CQ is defined as usual. The DP of whether, for a ∀∃ rule set $\mathbb{P}$ and a CQ Q, if $P \models_{\text{fol}} Q$ is called the ***CQ EP***, where $\models_{\text{fol}}$ is the standard first order logic entailment relation..

Notice that for any quad-graph $Q_\mathcal{C} = \{c_1\colon (s_1, p_1, o_1), \ldots, c_n\colon (s_r, p_r, o_r)\}$, let $r_{Q_\mathcal{C}}$ be the BR

$$\rightarrow \vec{\exists} y_{b_1}, \ldots, y_{b_q}\ c_1\colon (s_1, p_1, o_1)[\mu_\mathbf{B}] \wedge \ldots \wedge c_r\colon (s_r, p_r, o_r)[\mu_\mathbf{B}],$$

where $\{\_\colon b_1, \ldots, \_\colon b_q\}$ is the set of blank nodes in $Q_\mathcal{C}$, and $\mu_\mathbf{B}$ is the substitution function $\{\_\colon b_i \rightarrow y_{b_i}\}_{i=1,...,q}$ that assigns each blank-node to a fresh existentially quantified variable. It can be noted the quad-systems $\langle Q_\mathcal{C}, R\rangle$ and $\langle\emptyset, R \cup \{r_{Q_\mathcal{C}}\}\rangle$ are semantically equivalent.

The following property gives the relation between CCQ entailment of unrestricted quad-systems and standard first order CQ entailment of ∀∃ rule sets.

Property 6.1. *Suppose τ_q be the function from the set of quad patterns to the set of ternary atoms s.t. for any quad-pattern $c\colon (s,p,o)$, $\tau_q(c\colon (s,p,o)) = c(s,p,o)$.*

Let τ_{br} be a function from the set of BRs to the set of ∀∃ rules, s.t. for any BR r of the form (2):

$$\tau_{br}(r) = \forall\vec{x}\forall\vec{z}\,[\tau_q(c_1 : t_1(\vec{x},\vec{z})) \wedge ... \wedge \tau_q(c_n : t_n(\vec{x},\vec{z})) \rightarrow \exists\vec{y}\,\tau_q(c'_1 : t'_1(\vec{x},\vec{y})) \wedge ... \wedge \tau_q(c'_m : t'_m(\vec{x},\vec{y}))],$$

And, let τ be the function s.t. for any quad-system $QS_\mathcal{C} = \langle Q_\mathcal{C}, R\rangle$, $\tau(QS_\mathcal{C}) = \tau_{br}(R) \cup \{\tau_{br}(r_{Q_\mathcal{C}})\}$, where $\tau_{br}(R) = \bigcup_{r\in R}\tau_{br}(r)$.

Also, let τ_{ccq} be a function defined from the set of boolean CCQs to the set of boolean CQs, s.t. for any boolean CCQ $CQ = \exists\vec{y}\ c_1\colon t_1(\vec{a},\vec{y}) \wedge \ldots \wedge c_r : t_r(\vec{a},\vec{y})$, $\tau_{ccq}(CQ)$ is:

$$\exists\vec{y}\,\tau_q(c_1 : t_1(\vec{a},\vec{y})) \wedge \ldots \wedge \tau_q(c_r : t_r(\vec{a},\vec{y})),$$

then, for any quad-system $QS_\mathcal{C}$, CCQ CQ, $QS_\mathcal{C} \models CQ$ iff $\tau(QS_\mathcal{C}) \models_{fol} \tau_{ccq}(CQ)$.

Proof. Notice that every context $c \in \mathcal{C}$ becomes a ternary predicate symbol in the resulting translation. Also, $\tau(QS_\mathcal{C})$ is a $\forall\exists$ rule set, and for any CCQ CQ, $\tau_{ccq}(CQ)$ is a CQ.

In order to construct the restricted chase for $\tau(QS_\mathcal{C})$, suppose that $\prec_q$ is also extended to set of instances s.t. for any two quad-graphs $Q_\mathcal{C}$, $Q'_{\mathcal{C}'}$, $Q_\mathcal{C} \prec_q Q'_{\mathcal{C}'}$ iff $\tau_q(Q_\mathcal{C}) \prec_q \tau_q(Q'_{\mathcal{C}'})$. Suppose $\prec$ is extended similarly to set of instances. Also assume that during the construction of standard chase $chase(\tau(QS_\mathcal{C}))$ of $\tau(QS_\mathcal{C})$, for any application of a $\tau_{br}(r)$ with existentially quantified variables, with $r \in R$, suppose the skolem blank nodes generated in $chase(\tau(QS_\mathcal{C}))$ follow the same order as they are generated in $dChase(QS_\mathcal{C})$. Also let us extend the rule applicability function to the $\forall\exists$ rules settings s.t. for any set of BRs R, for any $r \in R$, quad-graph $Q'_{\mathcal{C}'}$, assignment μ, $applicable_R(r, \mu, Q'_{\mathcal{C}'})$ iff $applicable_{\tau_{br}(R)}(\tau_{br}(r), \mu, \tau_q(Q'_{\mathcal{C}'}))$.

Now $dChase_0(\langle\emptyset,\ R \cup \{r_{Q_\mathcal{C}}\}\rangle) = \emptyset$, and also $chase_0(\tau(QS_\mathcal{C})) = \emptyset$, and $dChase_1(QS_\mathcal{C}) = apply(r_{Q_\mathcal{C}}, \mu_\emptyset)$, where $\mu_\emptyset$ is the empty function, and $chase_1(\tau(QS_\mathcal{C}))$ $= apply(\tau_{br}(r_{Q_\mathcal{C}}),\ \mu_\emptyset)$, and so on. It is straightforward to see that for any $m \in \mathbb{N}$, $\tau_q(dChase_m(\langle\emptyset, R \cup \{r_{Q_\mathcal{C}}\}\rangle)) = chase_m(\tau(QS_\mathcal{C}))$. Hence, $\tau_q(dChase(QS_\mathcal{C}))$ $= chase(\tau(QS_\mathcal{C}))$, and $\{CQ\}[\sigma] \subseteq dChase(QS_\mathcal{C})$ iff $\{\tau_{ccq}(CQ)\}[\sigma] \subseteq chase(\tau(QS_\mathcal{C}))$.

Consequently, it follows that for any quad-system $QS_\mathcal{C} = \langle Q_\mathcal{C}, R\rangle$ and a boolean CCQ CQ, $QS_\mathcal{C} \models CQ$ iff $\tau(QS_\mathcal{C}) \models_{\text{fol}} \tau_{ccq}(CQ)$.

$\square$

Theorem 6.2. *There exists a polynomial time translation function τ (resp. τ_{ccq}) from the set of unrestricted quad-systems (resp. CCQs) to the set of $\forall\exists$ rule sets (resp. CQs), s.t. for any unrestricted quad-system $QS_\mathcal{C}$ and a CCQ CQ, $QS_\mathcal{C} \models CQ$ iff $\tau(QS_\mathcal{C})$ $\models_{fol} \tau_{ccq}(CQ)$.*

Proof. It is easy to see that τ_q, τ_{br}, τ, and τ_{ccq} in property 6.1 can be implemented using simple syntax transformation, by iterating through the respective components of a quad-system/CCQ, and the time complexity of these functions are linear w.r.t their inputs. $\square$

Notice that for any CCQ CQ (resp. CQ Q), $\rightarrow CQ$ (resp. $\rightarrow Q$) is a bridge (resp. $\forall\exists$) rule, with an empty body. Also, since for any quad-graph $Q_\mathcal{C}$, the translation function τ_{br} defined above can directly be applied on $r_{Q_\mathcal{C}}$ to obtain a $\forall\exists$ rule, the following theorem immediately follows:

Theorem 6.3. *For quad-systems, the EPs: (i) quad EP, (ii) quad-graph EP, (iii) BR EP, (iv) BRs EP, (v) Quad-System EP, and (vi) CCQ EP are polynomially reducible to entailment of $\forall\exists$ rule sets.*

A $\forall\exists$ rule set $\mathbb{P}$ is said to be a *ternary* $\forall\exists$ rule set, iff all the predicate symbols in the vocabulary of $\mathbb{P}$ is of arity less than or equal to three. $\mathbb{P}$ is a *purely ternary* rule set, iff all the predicate symbols in the vocabulary $\mathbb{P}$ is of arity three. Similarly, a (purely) ternary CQ is defined. The following property gives the relation between the CQ entailment problem of $\forall\exists$ rule sets and CCQ EP of unrestricted quad-systems.

Theorem 6.4. *There exists a polynomial time tranlation function ν (resp. ν_{cq}) from ternary $\forall\exists$ rule sets (resp. ternary CQs) to unrestricted quad-systems (resp. CCQs) s.t. for any $\forall\exists$ rule set $\mathbb{P}$ and a CQ Q, $\mathbb{P}$ $\models_{fol} CQ$ iff $\langle\emptyset,\ \nu(\mathbb{P})\rangle \models \nu_{cq}(Q)$.*

Proof. Note that the CQ EP of any ternary $\forall\exists$ rule set $\mathbb{P}$, whose set of predicate symbols is P, and CQ Q over P, can polynomially reduced to the CQ EP of a purely ternary rule set $\mathbb{P}'$ and purely ternary CQ Q', by the following transformation function χ. Let $\Box$ be an adhoc fresh URI; χ is s.t. for any ternary atom $c(s,p,o)$, $\chi(c(s,p,o)) = c(s,p,o)$. For any binary atom $c(s,p)$, $\chi(c(s,p)) = c(s,p,\Box)$, and for any unary atom $c(s)$, $\chi(c(s)) = c(s,\Box,\Box)$. For any $\forall\exists$ rule r of the form (7),

$$\chi(r) = \forall\vec{x}\forall\vec{z}\,[\chi(p_1(\vec{x},\vec{z})) \wedge \ldots \wedge \chi(p_n(\vec{x},\vec{z})) \rightarrow \exists\vec{y}\,\chi(p'_1(\vec{x},\vec{y})) \wedge \ldots \wedge \chi(p'_m(\vec{x},\vec{y}))]$$

And, for any $\forall\exists$ rule set $\mathbb{P}$, $\chi(\mathbb{P}) = \bigcup_{r\in\mathbb{P}} \chi(r)$. For any CQ Q, $\chi(Q)$ is similarly defined. Note that for any ternary $\forall\exists$ rule set $\mathbb{P}$, ternary CQ Q, $\chi(\mathbb{P})$ (resp. $\chi(Q)$) is purely ternary, and $\mathbb{P} \models_{\text{fol}} Q$ iff $\chi(\mathbb{P}) \models_{\text{fol}} \chi(Q)$.

Also, it can straightforwardly seen that $\tau_{br}^{-1}(\chi(\mathbb{P}))$ (resp. $\tau_{ccq}^{-1}(\chi(Q))$) is a set of BRs (resp. CCQ). Suppose, $\nu(\mathbb{P})$ is s.t. $\nu(\mathbb{P}) = QS_\mathcal{C} = \langle\emptyset,\ \tau_{br}^{-1}(\chi(\mathbb{P}))\rangle$. Intuitively, $\mathcal{C}$ contains a context identifier c, for each predicate symbol $c \in P$. Also suppose, $\nu_{cq}(Q) = \tau_{ccq}^{-1}(\chi(Q))$. Notice that $\nu_{cq}(Q)$ is CCQ. It can straightforwardly seen that ν and ν_{cq} can be computed in polynomial time, and $\mathbb{P} \models_{\text{fol}} Q$ iff $\nu(\mathbb{P}) \models \nu_{cq}(Q)$. $\square$

Thanks to the theorem 6.2 and theorem 6.4, the following theorem immediately holds:

Theorem 6.5. *The CCQ EP over quad-systems is polynomially equivalent to CQ EP over ternary $\forall\exists$ rule sets.*

By virtue of the theorem above, we derive the following property:

Property 6.6. *For quad-systems, the Quad EP, Quad-graph EP, BR(s) EP, and Quad-system EP are polynomially reducible to CCQ EP.*

Proof. The following claim is a folklore in the realm of $\forall\exists$ rules.

Claim (1) The $\forall\exists$ rule set EP is polynomially reducible to CQ EP.

Reducibility of $\forall\exists$ rule EP to CQ EP is a folklore in the realm of $\forall\exists$ rules. For a formal proof, we refer the reader to Baget et al. [14], where it is shown that the $\forall\exists$ rule EP is polynomially reducible to fact (a set of instances) EP, and fact EP are equivalent to CQ EP. Also, Cali et al [33] shows that CQ containment problem, which is equivalent to $\forall\exists$ rule EP, is reducible to CQ EP. Since a $\forall\exists$ rule set is a set of $\forall\exists$ rules, by using a series of oracle calls to a function that solves the $\forall\exists$ rule EP, we can define a function for deciding $\forall\exists$ rule set entailment. Hence, the claim holds.

(a) Thanks to translation functions τ, τ_{br} defined earlier, s.t. for any quad-system $QS_\mathcal{C}$, quad-graph $Q'_{\mathcal{C}'}$, $QS_\mathcal{C} \models Q'_{\mathcal{C}'}$ iff $\tau(QS_\mathcal{C}) \models_{\text{fol}} \tau_{br}(r_{Q'_{\mathcal{C}'}})$, we can infer that quad-graph EP is polynomially reducible to $\forall\exists$ rule set EP. Applying claim 1, it follows the quad-graph EP over quad-systems is polynomially reducible to CQ EP over $\forall\exists$ rule sets. By theorem 6.4, we can deduce that quad-graph EP is polynomially reducible to CCQ EP.

(b) By the translation functions τ and τ_{br}, defined earlier, s.t. for any quad-system $QS_\mathcal{C}$, a set of BRs R, $QS_\mathcal{C} \models R$ iff $\tau(QS_\mathcal{C}) \models_{\text{fol}} \tau_{br}(R)$, we can infer that BRs EP is polynomially reducible to $\forall\exists$ rule set EP. Similar to (a) above, we deduce that BRs EP is polynomially reducible to CCQ EP.

From (a) and (b), it follows that Quad-system EP is reducible to CCQ EP. $\square$

Having seen that the CCQ EP over quad-systems is polynomially equivalent to CQ EP over ternary $\forall\exists$ rule sets, we now compare some of the well known techniques used to ensure decidability of CQ entailment in the $\forall\exists$ rules settings to the decidability techniques for quad-systems that we saw earlier in the previous sections. Note that since all the quad-system classes we proposed in this paper are FECs, for a judicious comparison, the $\forall\exists$ rule classes to which we compare are classes which have a finite chase property. We compare to the following three well known classes: (i) Weakly Acyclic rule sets (WA), (ii) Jointly Acyclic rule sets (JA), and (iii) Model Faithful Acyclic $\forall\exists$ rule sets (MFA). The following property is well known in the realm of $\forall\exists$ rules:

Property 6.7. *For the any $\forall\exists$ rule set $\mathbb{P}$, the following holds:*

1. *If $\mathbb{P} \in$ WA, then $\mathbb{P} \in$ JA (from [36]),*
2. *If $\mathbb{P} \in$ JA, then $\mathbb{P} \in$ MFA (from [31]),*
3. *WA $\subset$ JA $\subset$ MFA (from [36] and [31]).*

Note that a description of few other $\forall\exists$ rule classes that do not have the finite chase property, but still enjoy decidability of CQ entailment are given in the related work.

6.1. Weak Acyclicity

Weak acyclicity [23,24] is a popular technique used to detect whether a $\forall\exists$ rule set has a finite chase, thus ensuring decidability of query answering. The set WA represents class of ternary $\forall\exists$ rule sets that have the weak acyclicity property.

For any predicate atom $p(t_1, \ldots, t_n)$, an expression $\langle p, i\rangle$, for $i = 1, \ldots, n$ is called a position of p. In the above case, t_1 is said to occur at position $\langle p, 1\rangle$, t_2 at $\langle p, 2\rangle$, and so on. For a set of $\forall\exists$ rules $\mathbb{P}$, its dependency graph is a graph whose nodes are positions of predicate atoms in $\mathbb{P}$; for each $r \in \mathbb{P}$ of the form (7), and for any variable x occurring in position $\langle p, i\rangle$ in head of r:

1. if x is universally quantified and x occurs in body of r at position $\langle p', j\rangle$, then there exists an edge from $\langle p', j\rangle$ to $\langle p, i\rangle$
2. if x is existentially quantified, then for any universally quantified variable x' occurring in head of r, with x' also occurring in the body of r at position $\langle p', j\rangle$, there exists a special edge from $\langle p', j\rangle$ to $\langle p, i\rangle$.

$\mathbb{P}$ is called weakly acyclic, iff its dependency graph does not contain cycles going through a special edge. For any $\forall\exists$ rule set $\mathbb{P}$, if $\mathbb{P}$ is WA, then its chase is finite, and hence CQ EP is decidable. Note that the nodes in the dependency graph that has incoming special edges corresponds to the positions of predicates where new values are created due to existential variables, and the normal edges capture the propagation of constants from one predicate position to another predicate position. In this way, absence of cycles involving special edges ensures that newly created Skolem

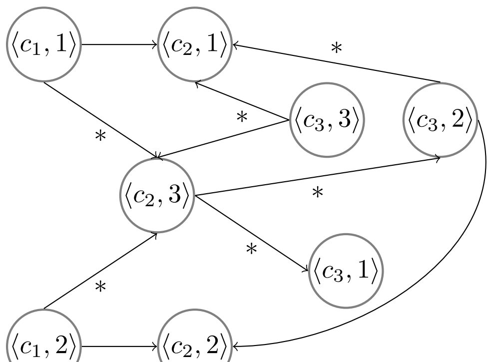

Fig. 3. dependency graph of the quad-system in example 4.3.

blank nodes are not recursively used to create other new Skolem blank nodes in the same position, leading to termination of chase computation.

Example 6.8. Let us revisit the quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}},\ R\rangle$ mentioned in example 4.3, whose dependency graph is shown in Fig. 3. Note that the $QS_{\mathcal{C}}$ is uncsafe, since its dChase contains a Skolem blank-node $\_\colon b_4$, which has as descendant another Skolem blank node $\_\colon b_1$, with the same origin context c_2 (see Fig. 1). However, it can be seen from Fig. 3 that the dependency graph of $\tau(QS_{\mathcal{C}})$ does not contain any directed cycle involving special edges. Hence, $\tau(QS_{\mathcal{C}})$ is weakly acyclic.

It turns out that there exists no inclusion relationship between the classes WA and CSAFE in either directions, i.e. WA $\not\subseteq$ CSAFE (from example 6.8), and CSAFE $\not\subseteq$ WA (from the fact that WA $\subset$ JA, and example 6.9 below). Whereas WA $\subset$ MSAFE, since WA $\subset$ MFA and MFA $\equiv$ MSAFE (theorem 6.10).

6.2. Joint Acyclicity

Joint acyclicity [36] extends weak acyclicity, by also taking into consideration the join between variables in body of $\forall\exists$ rules while analyzing the rules for acyclicity. The set JA represents the class of all ternary $\forall\exists$ rule sets that have the joint acyclicity property. A $\forall\exists$ rule set $\mathbb{P}$ is said to be *renamed apart*, if for any $r \neq r' \in R$, $\mathbf{V}(r) \cap \mathbf{V}(r') = \emptyset$. Since any set of rules can be converted to an equivalent renamed apart one by simple variable renaming, we assume that any rule set $\mathbb{P}$ is renamed apart. Also for any $r \in \mathbb{P}$ and for a variable y, let $Pos^r_H(y)$ ($Pos^r_B(y)$) be the set of positions in which y occurs in the head (resp. body) of r. For any $\forall\exists$ rule set $\mathbb{P}$ and an existentially quantified variable y occurring in a rule in $\mathbb{P}$, we define $Mov_{\mathbb{P}}(y)$ to be as follows:

- $Pos^r_H(y) \subseteq Mov_{\mathbb{P}}(y)$, if y occurs in r;
- $Pos^r_H(x) \subseteq Mov_{\mathbb{P}}(y)$, if x is a universally quantified variable and $Pos^r_B(x) \subseteq Mov_{\mathbb{P}}(y)$;

for any $r \in \mathbb{P}$. The existential dependency graph of a (renamed apart) set of rules $\mathbb{P}$ is a graph whose nodes are the existentially quantified variables in $\mathbb{P}$. There exists an edge from a variable y to y', if r is a rule in which y' occurs and there exists a universally quantified variable x in the head (and body) of r s.t. $Pos^r_B(x) \subseteq Mov_{\mathbb{P}}(y)$. A $\forall\exists$ rule set $\mathbb{P}$ is jointly acyclic, iff its existential dependency graph is acyclic. Analyzing the containment relationships, it happens to be the case that JA $\not\subseteq$ CSAFE (since WA $\subset$ JA, and eg. 6.8). Also example 6.9 shows us that CSAFE $\not\subseteq$ JA. However JA $\subset$ MSAFE, since JA $\subset$ MFA and MFA $\equiv$ MSAFE (theorem 6.10).

Example 6.9. Consider the quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$, where $Q_{\mathcal{C}} = \{c_1\colon (a, b, c)\}$. Suppose R is the following set:

$$R = \left\{ \begin{array}{lr} c_1\colon (x_{11}, x_{12}, z_1) \rightarrow c_2\colon (x_{11}, x_{12}, y_1) & (r_1) \\ c_1\colon (x_{21}, x_{22}, z_2), c_2\colon (x_{22}, x_{21}, x_{23}) \rightarrow & \\ c_3\colon (x_{21}, x_{22}, x_{23}) & (r_2) \\ c_3\colon (x_{31}, x_{32}, x_{33}) \rightarrow c_1\colon (x_{33}, x_{31}, x_{32}) & (r_3) \end{array} \right\}$$

Iterations during dChase construction are:

$$dChase_0(QS_{\mathcal{C}}) = \{c_1{:}(a, b, c)\}$$

$$dChase_1(QS_{\mathcal{C}}) = \{c_1\colon (a, b, c), c_2\colon (a, b, \_\colon b_1)\}$$

$$dChase(QS_{\mathcal{C}}) = dChase_1(QS_{\mathcal{C}})$$

Note that the lone Skolem blank node generated is $\_\colon b_1$, which do not have any descendants. Hence, by definition $QS_{\mathcal{C}}$ is csafe (msafe/safe). Now analyzing the BRs for joint acyclicity, we note that for the only existentially quantified variable y_1,

$Mov_R(y_1) = \{\langle c_2, 3\rangle, \langle c_3, 3\rangle, \langle c_1, 1\rangle\}$

Since the BR r_1 is which y_1 occurs contains the universally quantified variable x_{11} in head of r_1 s.t. $Pos^{r_1}_B(x_{11}) \subseteq Mov_R(y_1)$, there exists a cycle from y_1 to y_1 itself in the existential dependency graph of $\tau(QS_{\mathcal{C}})$. Hence, by definition $\tau(QS_{\mathcal{C}})$ is not joint acyclic. Also since the class of weakly acyclic rules are contained in the class of jointly acyclic rule, it follows that $\tau(QS_{\mathcal{C}})$ is also not weakly acyclic.

6.3. Model Faithful Acyclicity (MFA)

MFA, proposed in Bernardo et al. [31], is an acyclicity technique that guarantees finiteness of chase and decidability of query answering, in the realm of $\forall\exists$ rules. The set MFA denotes class of all ternary $\forall\exists$ rule sets that is model faithfully acyclic. As far as we know, the MFA technique subsumes almost all other known techniques that guarantee a finite chase, in the $\forall\exists$ rules settings. Obviously, WA $\subset$ JA $\subset$ MFA.

For any $\forall\exists$ rule $r = \phi(r)(\vec{x},\vec{z}) \rightarrow \psi(r)(\vec{x},\vec{y})$, for each $y_j \in \{\vec{y}\}$, let Y_r^j be a fresh unary predicate unique for y_j and r; furthermore, let S a be fresh binary predicate. The transformation **mfa** of r is defined as:

$$\mathbf{mfa}(r) = \phi(r)(\vec{x},\vec{z}) \rightarrow \psi(r)(\vec{x},\vec{y}) \wedge \bigwedge_{y_j\in\{\vec{y}\}} [Y_r^j(y_j) \wedge \bigwedge_{x_k\in\{\vec{x}\}} S(x_k,y_j)]$$

Also let r_1 and r_2 be two additional rules defined as:

$$S(x_1,z) \wedge S(z,x_2) \rightarrow S(x_1,x_2) \qquad (r_1)$$

$$Y_r^j(x_1) \wedge S(x_1,x_2) \wedge Y_r^j(x_2) \rightarrow \mathfrak{C} \qquad (r_2)$$

where $\mathfrak{C}$ is a fresh nullary predicate. For any set of $\forall\exists$ rules $\mathbb{P}$, let $ad(\mathbb{P})$ be the union of r_1 with the set of rules obtained by instantiating r_2, for each $r \in \mathbb{P}$, for each existential variable y_j in r. For a set of $\forall\exists$ rules $\mathbb{P}$, $\mathbf{mfa}(\mathbb{P}) = \bigcup_{r\in\mathbb{P}} \mathbf{mfa}(r) \cup ad(\mathbb{P})$. A $\forall\exists$ rule set $\mathbb{P}$ is said to be MFA, iff $\mathbf{mfa}(\mathbb{P}) \not\models_{\text{fol}} \mathfrak{C}$. It was shown in Cuenca Grau et al. [31] that if $\mathbb{P}$ is MFA, then $\mathbb{P}$ has a finite chase, thus ensuring decidability of query answering. The following theorem establishes the fact that notion of msafety is equivalent to MFA, thanks to the polynomial time translations between quad-systems and ternary $\forall\exists$ rule sets.

Theorem 6.10. *Let τ be the translation function from the set of unrestricted quad-systems to the set of ternary $\forall\exists$ rule sets, as defined in property 6.1, then, for any quad-system $QS_\mathcal{C} = \langle Q_\mathcal{C}, R\rangle$, $QS_\mathcal{C}$ is msafe iff $\tau(QS_\mathcal{C})$ is MFA.*

Proof. (outline) Recall that for $\tau = \langle\tau_q, \tau_{br}\rangle$, where τ_q is the quad translation function and τ_{br} is the translation function from BRs to $\forall\exists$ rules. Also, $\tau(QS_\mathcal{C}) = \tau_{br}(\{r_{Q_\mathcal{C}}\} \cup R)$. Also, recall that for every blank node b in $Q_\mathcal{C}$, the BR $r_{Q_\mathcal{C}}$ contains a corresponding existentially quantified variable y_b. We already saw that for such a transformation, the following property holds: for any $m \in \mathbb{N}$, $\tau_q(dChase_m(QS_\mathcal{C})) = chase_m(\tau(QS_\mathcal{C}))$, and for any BR $r \in R \cup \{r_{Q_\mathcal{C}}\}$, an assignment μ, $applicable_{R\cup\{r_{Q_\mathcal{C}}\}}(r,\ \mu,\ dChase_m(QS_\mathcal{C}))$ iff $applicable_{\tau(QS_\mathcal{C})}(\ \tau_{br}(r),\ \mu,\ chase_m(\tau(QS_\mathcal{C})))$. Also notice that for any two blank nodes $\_\!:b_1, \_\!:b_2$, $S(\_\!:b_1, \_\!:b_2) \in chase(\tau(QS_\mathcal{C}))$, iff $\_\!:b_1$ is a descendant of $\_\!:b_2$ in $dChase(QS_\mathcal{C})$. Hence, the relations S and descendantOf are identical.

Intuitively, MFA looks for cyclic creation of a skolem blank-node whose descendant is another skolem blank-node that are generated by the same rule $r = body(r)(\vec{x},\ \vec{z}) \rightarrow head(r)(\vec{x},\ \vec{y})$, by the same existential variable in $y_j \in \{\vec{y}\}$ of r. Wheras, msafety looks only for generation of a skolem blank-node $\_\!:b'$ whose descendant is another skolem $\_\!:b$ using the same rule r. Hence, if $\tau(QS_\mathcal{C})$ is not MFA, then $QS_\mathcal{C}$ is not msafe, and consequently onlyIf part of the theorem trivially holds.

(If part) Suppose $QS_\mathcal{C}$ is unmsafe, and μ and μ' are the assignments applied on $r \in R$ to create Skolem blank nodes $\_\!:b$ and $\_\!:b'$, respectively, and suppose $\_\!:b$ is a descendant of $\_\!:b'$ in the $dChase(QS_\mathcal{C})$. That is $\_\!:b = \mu(y_j)$ and $\_\!:b' = \mu'(y_k)$, for $y_j, y_k \in \{\vec{y}\}$ of r. Suppose $j = k$, then the prerequisite of non-MFA is trivially satisfied. Suppose if $j \neq k$ is the case, then there exists $\_\!:b''$ in $dChase(QS_\mathcal{C})$ s.t. $\_\!:b'' = \mu'(y_j)$, since μ' is applied on r and $y_j \in \{\vec{y}\}$. This means that also in this case, the prerequisite of non-MFA is satisfied. As a consequence $\tau(QS_\mathcal{C})$ is not MFA. Hence it follows that, $QS_\mathcal{C}$ is msafe iff $\tau(QS_\mathcal{C})$ is MFA. □

Let us revisit the quad-system $QS_\mathcal{C}$ in example 4.6, it can be easily seen that $\tau(QS_\mathcal{C})$ is not MFA. Recall that we have seen that $QS_\mathcal{C}$ is safe but not msafe. We consider the theorem 6.10 to be of importance, as it not only establishes the equivalence of MFA and msafety, but thanks to it and the translation τ, it can be deduced that the technique of safety, which we presented earlier, (strictly) extends the MFA technique. As far as we know, the MFA class of $\forall\exists$ rule sets is one of the most expressive class in the realm of $\forall\exists$ rule sets which allows a finite chase. Hence, the notion of safety that we propose can straightforwardly be ported to $\forall\exists$ settings. The main difference between MFA and safety is that MFA only looks for cyclic creation of two distinct Skolem blank-nodes $\_\!:b, \_\!:b'$ that are generated by the same rule r, by the same existential variable in r. Whereas safety also takes into account the origin vectors $\vec{a}$ and $\vec{a}'$ used during rule application to create $\_\!:b$ and $\_\!:b'$, respectively, and only raises an alarm if $\vec{a} \cong \vec{a}'$. Although, equivalence relation holds only between quad-systems and ternary $\forall\exists$ rule sets, it can easily be noticed that the technique of safety can be applied to $\forall\exists$ rule sets of arbitrary arity, and can be used to extend currently established tools and systems that work on existing notions of acyclicity such as WA, JA, or MFA.

7. Related Work

Contexts and Distributed Logics Work on contexts gained its attention as early as in the 80s, as 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 [17], *Distributed First Order Logics* by Ghidini et al. [16] and *Local Model Semantics* by Giunchiglia et al. [8]. 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., and *Context-OWL* [7] by Bouquet et al., and the work of CKR [12,9] 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/blank-node creation.

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

DL+rules Works on extending DL KBs with Datalog like rules was studied by Horrocks et al.[28] giving rise to the SWRL[28] language. The related initiatives proposes a formalism using which one can mix a DL ontology with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language, and hence enables horn-like rules to be combined with an OWL KB. Since SWRL is undecidable in general, studies on computable sub-fragments gave rise to works like Description Logic Rules [37], where the authors deal with rules that can be totally internalized by a DL knowledge base, and hence if the DL considered is decidable, then also is a DL+rules KB. The authors give various fragments of the rule bases like SROIQ rules, EL++ rules etc. and show that certain new constructs that are not expressible by plain DL can be expressed using rules although they are finally internalized into DL KBs. Unlike in our scenario, these works consider only horn rules with out existential variables.

$\forall\exists$ *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 [13] 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. These classes (according to [14]) can broadly be divided into the following three categories: (i) *bounded treewidth sets* (BTS), (ii) finite unification sets (FUS), and (iii) finite extension sets (FES). BTS contains the classes of $\forall\exists$ rule sets, whose models have bounded treewidth. Some of the important classes of these set are the linear $\forall\exists$ rules [19], (weakly) guarded rules [33], (weakly) frontier guarded rules [14], and jointly frontier guarded rules [36]. BTS classes in general need not have a finite chase, and query answering is done by exploiting the fact that the chase is tree shaped, whose nodes (which are sets of instances) start replicating (upto isomorphism) after a while. Hence, one could stop the computation of the chase, once it can be made sure that any future iterations of chase can only produce nodes that are isomorphic to existing nodes. A deterministic algorithm for deciding query entailment for this class is provided in Thomazo et al. [15].

FUS classes includes the class of 'sticky' rules [34], atomic hypothesis rules in which body of each rule contains only a single atom, and also the class of linear $\forall\exists$ rules. The approach used for query answering in FUS classes is to rewrite the input query w.r.t. to the $\forall\exists$ rule sets to another query that can be evaluated directly on the set of instances, s.t. the answers for the former query and latter query coincides. The approach is called the *query rewriting approach*. Compared to approaches proposed in this paper, these approaches do not enjoy the finite chase property, and is hence not conducive to materialization/forward chaining based query answering.

Unlike BTS and FUS, the FES classes are characterized by the finite chase property, and hence is most related to the techniques proposed in our work. Some of the classes in this set employ termination guarantying checks called 'acyclicity tests' that analyze the information flow between rules to check whether cyclic dependencies exists that can lead to infinite chase. *Weak acyclicity* [23,24], 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. The main drawback of these approaches is that they only analyze the schema/Tbox part of the rule sets, and ignore the instance part, and hence, produces a large number of false alarms, i.e. it is often the case that although dependency graph is cyclic, the chase is finite. Recently, a more dynamic approach, called the MFA technique, that also takes into account the instance part of the rule sets was proposed in Cuenca grau et al. [31], where existence of cyclic Skolem blank-node/constant generations in the chase is detected by augmenting the rules with extra information that keeps track of the Skolem function used to generate each Skolem blank-node. As shown in section 6, our technique of safety subsumes the MFA technique, and supports for much more expressive rule sets, by also keeping track of the vectors used by rule bodies while Skolem blank-nodes are generated.

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 [23], 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) [18], 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 [18], and decidability is attained by not allowing any kind cycles in the peer topology. Whereas in the context of Data exchange, WA is used in [23,24] to assure decidability, and the recent work by Marnette [35] employs the super weak acyclicity (SWA) to ensure decidability. It was shown in Cuenca Grau et al [31] that their MFA technique strictly subsumes both WA and SWA techniques in expressivity. Since, we saw in section 6 that our technique of safety subsumes the MFA technique and allows to represent much more expressive rule sets, safety technique can straightforwardly be employed in the above mentioned systems with decidability guarantees for query answering.

8. Summary and Conclusion

In this paper, we study the problem of query answering over contextualized RDF knowledge in the presence of forall-existential bridge rules. We show that the problem, in general, is undecidable, and present a few decidable classes of quad-systems. Table 1 displays the complexity results of chase computation and query entailment for the various classes of quad-systems, we have derived. Classes csafe, msafe, and safe, ensure decidability by restricting the structure of Skolem blank-nodes generated in the dChase. Briefly, the above classes do not allow an infinite descendant chain for Skolem blank-nodes generated, by constraining each Skolem blank-node in a descendant chain to have a different value for certain attributes, whose value sets are finite. RR and restricted RR quad-systems, do not allow the generation of Skolem blank nodes, thus constraining the dChase to have only constants from the initial quad-system. The above classes which suit varying situations, can be used to extend the currently established tools for contextual reasoning to give support for expressive bridge rules with conjunctions and existential quantifiers with decidability guarantees. From an expressivity point of view, the class of safe quad-systems subsumes all the above classes, and other well known classes in the realm of $\forall\exists$ rules with finite chases. We view the results obtained in this paper as a general foundation for contextual reasoning and query answering over contextualized RDF knowledge formats such as quads, and can straightforwardly be used to extend existing quad stores.

Quad-System Fragment	Chase size w.r.t input quad-system	Data Complexity of CCQ entailment	Combined Complexity of CCQ entailment
Unrestricted Quad-Systems	Infinite	Undecidable	Undecidable
Safe Quad-Systems	Double exponential	PTIME-complete	2EXPTIME-complete
MSafe Quad-Systems	Double exponential	PTIME-complete	2EXPTIME-complete
CSafe Quad-Systems	Double exponential	PTIME-complete	2EXPTIME-complete
RR Quad-Systems	Polynomial	PTIME-complete	EXPTIME-complete
Restricted RR Quad-Systems	Polynomial	PTIME-complete	NP-complete

Table 1

Complexity info for various quad-system fragments

References

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

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

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

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

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

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

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

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

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

[10] M.Joseph and G.Kuper and L.Serafini. Query Answering over contextualized RDF Knowledge with Forall-Existential Bridge Rules: Attaining Decidability using Acyclicity. In *Proc. of Italian Conference in Computational Logic (CILC-2014)*, 2014.

[11] M.Joseph and G.Kuper and L.Serafini. Query Answering over contextualized RDF/OWL Knowledge with Forall-Existential Bridge Rules: Attaining Decidability using Acyclicity. In *Proc. of International Conference on Web Reasoning and Rule Systems (RR-2014), To Appear*, 2014.

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

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

[14] Baget, J.-F.; Leclère, M.; Mugnier, M.-L.; and Salvat, E. 2011. On rules with existential variables: Walking the decidability line. *Artificial Intelligence* 175(9-10):1620–1654.

[15] Thomazo, M.; Baget, J.-F.; Mugnier, M.-L.; and Rudolph, S. A Generic Querying Algorithm for Greedy Sets of Existential Rules. In *KR'12: International Conference on Principles of Knowledge Representation and Reasoning*, 096–106. 2012.

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

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

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

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

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

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

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

[23] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa, "Data Exchange: Semantics and Query Answering," in *Theoretical Computer Science*, pp. 28(1):89–124, 2005.

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

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

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

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

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

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

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

[31] B. Cuenca Grau, I. Horrocks, M. Krötzsch, C. Kupke, D. Magka, B. Motik, and Z. Wang, "Acyclicity Notions for Existential Rules and Their Application to Query Answering in Ontologies," in *Journal of Artificial Intelligence Research (JAIR)*, vol. 47, pp. 741–808, AI Access Foundation, 2013.

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

[33] A. Calì, G. Gottlob, and M. Kifer Taming the infinite chase: Query answering under expressive relational constraints. In *KR*, 70–80. 2008.

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

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

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

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

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

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

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

Appendix

A. Proofs of Section 3

Property 3.1. Note that a strict linear order is a relation that is irreflexive, transitive, and linear.

Irreflexivity: By contradiction, suppose $\prec_q$ is not irreflexive, then there exists $Q \in \mathcal{Q}$ s.t. $Q \prec_q Q$ holds. This means that neither of the conditions (i) and (ii) of $\prec_q$ definition holds for Q. Hence, due to condition (iii) $Q \not\prec_q Q$, which is a contradiction.

Linearity: Note that for any two distinct $Q, Q' \in \mathcal{Q}$, one of the following holds: (a) $Q \subset Q'$, (b) $Q' \subset Q$, or (c) $Q \setminus Q'$ and $Q' \setminus Q$ are non-empty and disjoint. Suppose (a) is the case, then $Q \prec_q Q'$ holds. Similarly, if (b) is the case then $Q' \prec_q Q$ holds. Otherwise if (c) is the case, then by condition (ii), either $Q \prec_q Q'$ or $Q' \prec_q Q$ should hold. Hence, $\prec_q$ is a linear order over $\mathcal{Q}$.

Transitivity: Suppose there exists $Q, Q', Q'' \in \mathcal{Q}$ s.t. $Q \prec_q Q'$ and $Q' \prec_q Q''$. Then, one of the following four cases hold: (a) $Q \prec_q Q'$ due to (i) and $Q' \prec_q Q''$ due to (i), (b) $Q \prec_q Q'$ due to (i) and $Q' \prec_q Q''$ due to (ii), (c) $Q \prec_q Q'$ due to (ii) and $Q' \prec_q Q''$ due to (i), (d) $Q \prec_q Q'$ due to (ii) and $Q' \prec_q Q''$ due to (ii).

Suppose if (a) is the case, then trivially $Q \subset Q''$, and hence by applying condition (i) $Q \prec_q Q''$. Otherwise if (b) is the case, then either (1) $Q \subset Q''$ or (2) $Q \not\subset Q''$. Suppose, (1) is the case then, by (i) $Q \prec_q Q''$. Otherwise, if (2) is the case, then since, $Q \subset Q'$, it cannot be the case that greatestQuad$_{\prec_l}(Q'' \setminus Q) \prec_l$ greatestQuad$_{\prec_l}(Q'' \setminus Q')$, and it cannot be the case that greatestQuad$_{\prec_l}(Q' \setminus Q'') \prec_l$ greatestQuad$_{\prec_l}(Q \setminus Q'')$. Hence, it should be the case that greatestQuad$_{\prec_l}(Q'' \setminus Q') \preceq_l$ greatestQuad$_{\prec_l}(Q'' \setminus Q)$ and greatestQuad$_{\prec_l}(Q \setminus Q'') \prec_l$ greatestQuad$_{\prec_l}(Q' \setminus Q'')$. But since, greatestQuad$_{\prec_l}(Q' \setminus Q'') \prec_l$ greatestQuad$_{\prec_l}(Q'' \setminus Q')$, it follows that greatestQuad$_{\prec_l}(Q \setminus Q'') \prec_l$ greatestQuad$_{\prec_l}(Q'' \setminus Q)$, and hence by condition (ii), $Q \prec_q Q''$. Hence, if (b) is the case, then in both possible case (1) or (2), it should be the case that $Q \prec_q Q''$. Otherwise if (c) is the case, then similar to the arguments in (b), by condition (i) or (ii), it can easily be the seen that $Q \prec_q Q''$.

Otherwise, if (d) is the case, then it should be the case that greatestQuad$_{\prec_l}(Q \setminus Q') \prec_l$ greatestQuad$_{\prec_l}(Q' \setminus Q)$ (†) and greatestQuad$_{\prec_l}(Q' \setminus Q'') \prec_l$ greatestQuad$_{\prec_l}(Q'' \setminus Q')$ (‡). Suppose by contradiction $Q'' \prec_q Q$, then one of the following holds: (1) $Q'' \prec_q Q$ by condition (i) or (2) $Q'' \prec_q Q$ by condition (ii). Suppose, (1) is the case, then it should be the case that $Q'' \subset Q$. Hence, it should not be the case that greatestQuad$_{\prec_l}(Q \setminus Q') \prec_l$ greatestQuad$_{\prec_l}(Q'' \setminus Q')$ and it should not be the case that greatestQuad$_{\prec_l}(Q' \setminus Q'') \prec_l$ greatestQuad$_{\prec_l}(Q' \setminus Q)$. Hence, it should be the case that greatestQuad$_{\prec_l}(Q'' \setminus Q') \preceq_l$ greatestQuad$_{\prec_l}(Q \setminus Q')$ (♡), and it should be the case that greatestQuad$_{\prec_l}(Q' \setminus Q) \preceq_l$ greatestQuad$_{\prec_l}(Q' \setminus Q'')$ (♠). Applying, (‡) in (♡), we get greatestQuad$_{\prec_l}(Q' \setminus Q'') \prec_l$ greatestQuad$_{\prec_l}(Q \setminus Q')$, and Applying, (†) in (♠), we get greatestQuad$_{\prec_l}(Q \setminus Q') \prec_l$ greatestQuad$_{\prec_l}(Q' \setminus Q'')$, which is a contradiction. Suppose if (2) is the case, then greatestQuad$_{\prec_l}(Q'' \setminus Q) \prec_l$ greatestQuad$_{\prec_l}(Q \setminus Q'')$. The above can be written as: greatestQuad$_{\prec_l}(Q'' \setminus (Q \cap Q'')) \prec_l$ greatestQuad$_{\prec_l}(Q \setminus (Q \cap Q''))$. Using $Q \cap Q' \cap Q'' \subseteq Q \cap Q'$, it follows that greatestQuad$_{\prec_l}(Q''\setminus(Q\cap Q'\cap Q'')) \preceq_l$ greatestQuad$_{\prec_l}(Q \setminus (Q \cap Q' \cap Q''))$ (♣). Also applying similar transformation in (†) and (‡), we get greatestQuad$_{\prec_l}(Q \setminus (Q \cap Q' \cap Q'')) \preceq_l$ greatestQuad$_{\prec_l}(Q' \setminus (Q \cap Q' \cap Q''))$, and greatestQuad$_{\prec_l}(Q' \setminus (Q \cap Q' \cap Q'')) \preceq_l$ greatestQuad$_{\prec_l}(Q'' \setminus (Q \cap Q' \cap Q''))$. From which, it follows that greatestQuad$_{\prec_l}(Q \setminus (Q \cap Q' \cap Q''))$ $\preceq_l$ greatestQuad$_{\prec_l}(Q'' \setminus (Q \cap Q' \cap Q''))$. Using (♣) in the above, we get greatestQuad$_{\prec_l}(Q \setminus (Q \cap Q' \cap Q''))$ = greatestQuad$_{\prec_l}(Q' \setminus (Q \cap Q' \cap Q''))$ = greatestQuad$_{\prec_l}(Q'' \setminus (Q \cap Q' \cap Q''))$, which is a contradiction. Hence, it should be the case that $Q \prec_q Q''$. □

Theorem 3.5. We show that CCQ entailment is undecidable for unrestricted quad-systems, by showing that the well known undecidable problem of "non-emptiness of intersection of context-free grammars" is reducible to the CCQ answering problem.

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

$$\vec{w} \rightarrow \vec{w}'$$

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:

Property A.1. *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:

$$c\colon (x_1, w_1, x_2), c\colon (x_2, w_2, x_3), ..., c\colon (x_n, w_n, x_{n+1}) \rightarrow c\colon (x_1, v, x_{n+1}) \quad (9)$$

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

$$c\colon (x, \texttt{rdf:type}, C) \rightarrow \exists y\, c\colon (x, t_i, y), c\colon (y, \texttt{rdf:type}, C) \quad (10)$$

and Q_c contains only the triple:

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

We in the following show that:

$$QS_c \models \exists y\, c\colon (a, S_1, y) \wedge c\colon (a, S_2, y) \leftrightarrow L(G_1) \cap L(G_2) \neq \emptyset \quad (11)$$

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

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

base case suppose if $|\vec{w}| = 1$, then $\vec{w} = t_i$, for some $t_i \in T$. But Since by construction $c\colon (a, \texttt{rdf:type}, C) \in dChase_0(QS_c)$, on which rules of the form (10) is applicable. Hence, there exists an i such that $dChase_i(QS_c)$ contains $c\colon (a, t_i, b_i)$, $c\colon (b_i, \texttt{rdf:type}, C)$, for each $t_i \in T$. Hence, the base case.

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

inductive step suppose $\vec{w} = t_1...t_{p+1}$, with $|\vec{w}| \leq p' + 1$. Since $\vec{w}$ can be written as $\vec{w'}t_{p+1}$, where $\vec{w}' = t_1...t_p$, and by hypothesis, there exists $b_1, ..., b_p$ such that $c\colon (a, t_1, b_1)$, $c\colon (b_1, t_2, b_2)$, ..., $c\colon (b_{p-1}, t_p, b_p)$, $c\colon (b_p, \texttt{rdf:type}, C) \in$

$dChase(QS_c)$. Also since rules of the form (10) are applicable on c: (b_p, `rdf:type`, C), and hence produces triples of the form c: (b_p, t_i, b^i_{p+1}), c: (b^i_{p+1}), `rdf:type`, C), for each $t_i \in T$. Since $t_{p+1} \in T$, the claim follows.

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

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

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

base case Suppose $|\vec{w}| = 1$, then $\vec{w} = t_k$, for some $t_k \in T$. If there exists b_1, b_2 such that c: (b_1, t_k, b_2). But since there exists a PR $V_j \rightarrow t_k$, by transformation given in (9), there exists a BR c: $(x_1, t_k, x_2) \rightarrow c$: $(x_1, V_j, x_2) \in R$, which is applicable on c: (b_1, t_k, b_2) and hence the quad c: $(b_1, V_j, b_2) \in dChase(QS_c)$.

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

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

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

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

(a) For any $\vec{w} = t_1...t_p \in T^*$, if $\vec{w} \in L(G_1) \cap L(G_2)$, then by claim 1, since there exists $b_1, ..., b_p$, such that c: $(a, t_1, b_1), ..., c$: $(b_{p-1}, t_p, b_p) \in dChase(QS_c)$. But since $\vec{w} \in L(G_1)$ and $\vec{w} \in L(G_2)$, $S_1 \rightarrow \vec{w}$ 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 $dChase(QS_c)$ $\models \exists y\ c$: $(a, s_1, y) \wedge c$: (a, s_2, y). Hence, $QS_c \models \exists y$ c: $(a, s_1, y) \wedge c$: (a, s_2, y).
(b) Suppose if $QS_c \models \exists y\ c$: $(a, S_1, y) \wedge c$: (a, S_2, y), then this implies that there exists b_p such that c: (a, S_1, b_p), c: $(a, S_2, b_p) \in dChase(QS_\mathcal{C})$. Then it is the case that there exists $\vec{w} = t_1...t_p \in T^*$, and $b_1, ..., b_p$ such that c: (a, t_1, b_1), ..., c: (b_{p-1}, t_p, b_p), c: (a, S_1, b_p), c: $(a, S_2, b_p) \in dChase(QS_c)$. Then by claim 3, $S_1 \rightarrow^* \vec{w}$, $S_2 \rightarrow^* \vec{w}$. Hence, $w \in L(G_1) \cap L(G_2)$.

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

B. Proofs of Section 4

Theorem 4.13. We in the following show the case of $dChase^{\text{csafe}}(QS_\mathcal{C})$, i.e. **unCSafe** $\in dChase^{\text{csafe}}(QS_\mathcal{C})$ iff $QS_\mathcal{C}$ is uncsafe. The proof follows from lemma B.1 and lemma B.2 below.

The proofs for the case of $dChase^{\text{safe}}(QS_\mathcal{C})$ and $dChase^{\text{msafe}}(QS_\mathcal{C})$ is similar, and is omitted.

□

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

Proof. Note that $augC(R) = \bigcup_{r \in R} augC(r) \cup \{brTR\}$, where $brTR$ is the range restricted BR c_c: (x_1, descendantOf, z), c_c: (z, descendantOf, x_2) $\rightarrow c_c$: (x_1, descendantOf, x_2). Also for each $r \in R$,

$body(r) = body(augC(r))$, and for any $c \in \mathcal{C}$, $c\colon (s,p,o) \in head(r)$ iff $c\colon (s,p,o) \in head(augC(r))$. That is, $head(r) = head(augC(r))(\mathcal{C})$, where $head(r)(\mathcal{C})$ denotes the quad-patterns in $head(r)$, whose context identifiers is in $\mathcal{C}$. Also, $head(augC(r)) = head(augC(r))(\mathcal{C}) \cup head(augC(r))(c_c)$, and also the set of existentially quantified variables in $head(augC(r))(c_c)$ is contained in the set of existentially quantified variables in $head(augC(r))(\mathcal{C})$ (†). We first prove the following claim:

Claim (0) For any quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$, let i be a csafe dChase iteration, let j be the number of csafe dChase iterations before i in which $brTR$ was applied, then $dChase_{i-j}(QS_{\mathcal{C}}) = dChase_i^{\text{csafe}}(QS_{\mathcal{C}})(\mathcal{C})$.

We approach the proof the above claim by induction on i.

base case If $i = 1$, then $dChase_0^{\text{csafe}}(QS_{\mathcal{C}})(c_c) = \emptyset$ and $dChase_0^{\text{csafe}}(QS_{\mathcal{C}})(\mathcal{C}) = dChase_0^{\text{csafe}}(QS_{\mathcal{C}}) = dChase_0(QS_{\mathcal{C}})$. Hence, it should be the case that $applicable_{augC(R)}(brTR, \mu, dChase_0^{\text{csafe}}(QS_{\mathcal{C}}))$ does not hold, for any μ. Hence, $applicable_R(r, \mu, dChase_0(QS_{\mathcal{C}}))$ iff $applicable_{augC(R)}(augC(r), \mu, dChase_0^{\text{csafe}}(QS_{\mathcal{C}}))$, for any $r \in R$, assignment μ. Also using (†), it follows that $dChase_1(QS_{\mathcal{C}}) = dChase_{1-0}^{\text{csafe}}(QS_{\mathcal{C}})(\mathcal{C})$.

hypothesis for any $i \leq k$, if i is a csafe dChase iteration, and j be the number of csafe dChase iterations before i in which $brTR$ was applied, then $dChase_{i-j}(QS_{\mathcal{C}}) = dChase_i^{\text{csafe}}(QS_{\mathcal{C}})(\mathcal{C})$.

inductive suppose $i = k+1$, then one of the following three cases should hold: (a) $applicable_{augC(R)}(r, \mu, dChase_k^{\text{csafe}}(QS_{\mathcal{C}}))$ does not hold for any $r \in augC(R)$, assignment μ, and $dChase_{k+1}^{\text{csafe}}(QS_{\mathcal{C}}) = dChase_k^{\text{csafe}}(QS_{\mathcal{C}})$, or (b) $applicable_{augC(R)}(brTR, \mu, dChase_k^{\text{csafe}}(QS_{\mathcal{C}}))$ holds, for some assignment μ, or (c) $applicable_{augC(R)}(r, \mu, dChase_k^{\text{csafe}}(QS_{\mathcal{C}}))$ holds, for some $r \in augC(R) \setminus \{brTR\}$, for some assignment μ. If (a) is the case, then it should be the case that $applicable_R(r', \mu, dChase_{k-j}(QS_{\mathcal{C}}))$ does not hold, for any $r' \in R$, assignment μ. As a result $dChase_{k+1-j}(QS_{\mathcal{C}}) = dChase_{k-j}(QS_{\mathcal{C}})$, and hence, $dChase_{k+1-j}(QS_{\mathcal{C}}) = dChase_{k+1}^{\text{csafe}}(QS_{\mathcal{C}})(\mathcal{C})$. If (b) is the case, then since $dChase_{k+1}^{\text{csafe}}(QS_{\mathcal{C}})(\mathcal{C}) = dChase_k^{\text{csafe}}(QS_{\mathcal{C}})(\mathcal{C})$, $dChase_{k+1}^{\text{csafe}}(QS_{\mathcal{C}})(\mathcal{C}) = dChase_{k+1-j-1}(QS_{\mathcal{C}}) = dChase_{k-j}(QS_{\mathcal{C}})$. If (c) is the case, then it should the case that $applicable_R(r', \mu, dChase_{k-j}(QS_{\mathcal{C}})$, where $r = augC(r')$ and $head(r)(\mathcal{C}) = head(r)$. Hence, it should be the case that $dChase_{k+1}^{\text{csafe}}(QS_{\mathcal{C}})(\mathcal{C}) = dChase_{k+1-j}(QS_{\mathcal{C}})$.

The following claim, which straightforwardly follows from claim 0, shows that any quad $c\colon (s,p,o)$, with $c \in \mathcal{C}$ derived in csafe dChase, is also derived in its standard dChase. In this way, csafe dChase do not generate any unsound triples in any context $c \in \mathcal{C}$.

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

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

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

If $c_c\colon (b$, originContext, $c) \in dChase^{\text{csafe}}(QS_{\mathcal{C}})$, there exists $i \in \mathbb{N}$, s.t. $c_c\colon (b$, originContext, $c) \in dChase_i^{\text{csafe}}(QS_{\mathcal{C}})$ and there exists no $j < i$ with $c_c\colon (b$, originContext, $c) \in dChase_j^{\text{csafe}}(QS_{\mathcal{C}})$. But if $c_c\colon (b$, originContext, $c) \in dChase_i^{\text{csafe}}(QS_{\mathcal{C}})$ implies that there exists an $augC(r) = body(\vec{x}, \vec{z}) \rightarrow head(\vec{x}, \vec{y}) \in augC(R)$, with $c_c\colon (y_j$, originContext, $c) \in head(\vec{x}, \vec{y})$, $y_j \in \{\vec{y}\}$, s.t. $c_c\colon (b$, originContext, $c)$ was generated due to application of an assignment μ on $augC(r)$, with $b = y_j[\mu^{ext(\vec{y})}]$. This implies that there exists $c\colon (s,p,o) \in head(\vec{x},\vec{y})$, with $s = y_j$ or $p = y_j$ or $o = y_j$, $c \in \mathcal{C}$. Since according to our assumption, i is the first iteration in which $c_c\colon (b$, originContext, $c)$ is generated, it follows that i is the first iteration in which $c\colon (s,p,o)[\mu^{ext(\vec{y})}]$ is also generated. Let k be the number of iterations before i in which $brTR$ was applied. By applying claim 0, it should be the case that $c\colon (s,p,o)[\mu^{ext(\vec{y})}] \in dChase_{i-k}(QS_{\mathcal{C}})$, and $i - k$ should be the first such dChase iteration. Hence, $c \in orginContexts(b)$.

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

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

Since any quad of the form $c_c\colon (b'$, descendantOf, $b) \in dChase^{\text{csafe}}(QS_{\mathcal{C}})$ is not an element of $Q_{\mathcal{C}}$, and can only be introduced by an application of a BR $r \in augC(R)$, any quad of the form $c_c\colon (b'$, descendantOf, $b)$ can only be introduced, earliest in the first iteration of $dChase^{\text{csafe}}(QS_{\mathcal{C}})$. Suppose $c_c\colon (b'$, descendantOf, $b) \in dChase^{\text{csafe}}(QS_{\mathcal{C}})$, then there exists an iteration $i \geq 1$ s.t. $c_c\colon (b'$, descendantOf, $b)$

$\in dChase_j^{\text{csafe}}(QS_{\mathcal{C}})$, for any $j \geq i$, and c_c: (b', descendantOf, b) $\notin dChase_{j'}^{\text{csafe}}(QS_{\mathcal{C}})$, for any $j' < i$. We apply induction on i for the proof.

base case suppose c_c:(b', descendantOf, b) $\in dChase_1^{\text{csafe}}(QS_{\mathcal{C}})$ and since $b \neq b'$, then there exists a BR $r \in augC(R)$, $\exists\mu$ s.t. $applicable_{augC(R)}(r, \mu, dChase_0^{\text{csafe}}(QS_{\mathcal{C}}))$, i.e. $body(r)(\vec{x}, \vec{z})[\mu] \subseteq dChase_0^{\text{csafe}}(QS_{\mathcal{C}})$ and c_c: (b', descendantOf, b) $\in head(r)(\vec{x}, \vec{y})[\mu^{ext(\vec{y})}]$. Then by construction of $augC(r)$, it follows that $b = y_j[\mu^{ext(\vec{y})}]$, for some $y_j \in \{\vec{y}\}$ and $b' = \mu(x_i)$, for some $x_i \in \{\vec{x}\}$. Since $dChase_0(QS_{\mathcal{C}}) = dChase_0^{\text{csafe}}(QS_{\mathcal{C}})$, it follows using (†) that $applicable_R(r', \mu, dChase_0(QS_{\mathcal{C}}))$ holds, for $r' = body(r')(\vec{x}, \vec{z}) \to head(r')(\vec{x}, \vec{y})$, with $augC(r') = r$. Hence, by construction, it follows that $b = y_j[\mu^{ext(\vec{y})}] \in \mathbf{C}(dChase_1(QS_{\mathcal{C}}))$, for $y_j \in \{\vec{y}\}$ and $b' = \mu(x_i)$, for $x_i \in \{\vec{x}\}$. Hence b' is a descendant of b (by definition).

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

inductive step suppose c_c: (b', descendantOf, b) $\in dChase_{k+1}^{\text{csafe}}(QS_{\mathcal{C}})$, then either (i) c_c: (b', descendantOf, b) $\in dChase_k^{\text{csafe}}(QS_{\mathcal{C}})$ or (ii) c_c: (b', descendantOf, b) $\notin dChase_k^{\text{csafe}}(QS_{\mathcal{C}})$. Suppose (i) is the case, then by hypothesis, b' is a descendant of b. If (ii) is the case, then either (a) c_c: (b', descendantOf, b) is the result of the application of a $brTR \in augC(R)$ on $dChase_k^{\text{csafe}}(QS_{\mathcal{C}})$ or (b) c_c: (b', descendantOf, b) is the result of the application of a $r \in augC(R) \setminus \{brTR\}$ on $dChase_k^{\text{csafe}}(QS_{\mathcal{C}})$. If (a) is the case, then there exists a $b'' \in \mathbf{C}(dChase_k^{\text{csafe}}(QS_{\mathcal{C}}))$ s.t. c_c: (b', descendantOf, b'') $\in dChase_k^{\text{csafe}}(QS_{\mathcal{C}})$ and c_c: (b'', descendantOf, b) $\in dChase_k^{\text{csafe}}(QS_{\mathcal{C}})$. Hence, by hypothesis b' is a descendantOf b'' and b'' is a descendantOf b. Since 'descendantOf' relation is transitive, b' is a descendantOf b. Otherwise if (b) is the case then similar to the arguments used in the base case, it can easily be seen that b' is a descendant of b.

Suppose if the quad **unCSafe** $\in dChase^{\text{csafe}}(QS_{\mathcal{C}})$, then this implies that there exists an iteration i s.t. the function unCSafeTest on $augC(r)$, with $r = body(r)(\vec{x}, \vec{z}) \to head(r)(\vec{x}, \vec{y}) \in R$, assignment μ, and $dChase_i^{\text{csafe}}(QS_{\mathcal{C}})$ returns True. This implies that, there exists $b, b' \in \mathbf{B}$, $y_j \in \{\vec{y}\}$ s.t. $body(r)(\vec{x}, \vec{z})[\mu] \subseteq dChase_i^{\text{csafe}}(QS_{\mathcal{C}})$, $b \in \{\mu(\vec{x})\}$, c_c: (b', descendantOf, b) $\in dChase_i^{\text{csafe}}(QS_{\mathcal{C}})$ and $\{c \mid c_c$: (b', originContext, c) $\in dChase_i^{\text{csafe}}(QS_{\mathcal{C}})\} = cScope(y_j, head(r)(\vec{x}, \vec{y}))$. Suppose k be the number of csafe dChase iterations before i, in which $brTR$ was applied. Hence, by claim 0, $dChase_{i-k-1}(QS_{\mathcal{C}}) = dChase_{i-1}^{\text{csafe}}(QS_{\mathcal{C}})(\mathcal{C})$, and consequently $applicable_R(r, \mu, dChase_{i-k-1}(QS_{\mathcal{C}}))$ holds. Hence, as a result of μ being applied on r, there exists $b'' = y_j[\mu^{ext(\vec{y})}] \in \mathbf{B}(dChase_{i-k}(QS_{\mathcal{C}})))$, with $b \in \{\mu(\vec{x})\}$. Hence, by definition $originContext(b'') = cScope(y_j, head(r))$, and b is a descendantOf b''. If $b \neq b'$, then by Claim 2, b' is a descendantOf b, otherwise $b' = b$ and hence b' is a descendantOf b''. Consequently, b' is a descendantOf b''. Also, applying claim 3, we get that $originContexts(b') = originContexts(b'')$, which means that prerequisites of uncsafety is satisfied, and hence, $QS_{\mathcal{C}}$ is uncsafe. □

Lemma B.2 (Completeness)**.** *For any quad-system, $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$, if $QS_{\mathcal{C}}$ is uncsafe then **unCSafe** $\in dChase^{csafe}(QS_{\mathcal{C}})$.*

Proof. We first prove a few supporting claims in order to prove the theorem.

Claim (0) For any quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$, suppose **unCSafe** $\notin dChase^{\text{csafe}}(QS_{\mathcal{C}})$, then for any dChase iteration i, there exists a $j \geq 0$ s.t. $dChase_i(QS_{\mathcal{C}}) = dChase_{i+j}^{\text{csafe}}(QS_{\mathcal{C}})(\mathcal{C})$.

We approach the proof by induction on i.

base case for $i = 0$, we know that $dChase_0(QS_{\mathcal{C}}) = dChase_0^{\text{csafe}}(QS_{\mathcal{C}}) = Q_{\mathcal{C}}$. Hence, the base case trivially holds.

hypothesis for $i \leq k \in \mathbb{N}$, there exists $j \geq 0$ s.t. $dChase_i(QS_{\mathcal{C}}) = dChase_{i+j}^{\text{csafe}}(QS_{\mathcal{C}})$

step case for $i = k + 1$, one of the following holds: (a) $dChase_{k+1}(QS_{\mathcal{C}}) = dChase_k(QS_{\mathcal{C}})$ or (b) $dChase_{k+1}(QS_{\mathcal{C}}) = dChase_k(QS_{\mathcal{C}}) \cup head(r)(\vec{x}, \vec{y})[\mu^{ext(\vec{y})}]$ and $applicable_R(r, \mu, dChase_k(QS_{\mathcal{C}}))$ holds, for some $r = body(r)(\vec{x}, \vec{z}) \to head(r)(\vec{x}, \vec{y})$, assignment μ. If (a) is the case, then trivially the claim holds. Otherwise, if (b) is the case, then let $j \in \mathbb{N}$ be s.t. $dChase_k(QS_{\mathcal{C}}) = dChase_{k+j}^{\text{csafe}}(QS_{\mathcal{C}})(\mathcal{C})$. Let $j' \geq j, l \in \mathbb{N}$ be s.t. $applicable_{augC(R)}(brTR, \mu, dCase_{k+l}^{\text{csafe}}(QS_{\mathcal{C}}))$, for any $j' \geq l \geq j$, and $applicable_{augC(R)}(brTR, \mu, dCase_{k+j'+1}^{\text{csafe}}(QS_{\mathcal{C}}))$ does not hold. By construction, it should be the case that $applicable(r', \mu, dCase_{k+j'+1}^{\text{csafe}}(QS_{\mathcal{C}}))$ holds, where $r' = augC(r)$. Also since no new Skolem blank node was introduced in any csafe dChase iteration $k + l$, for any $j \leq l \leq j'$. It should be the case that $head(r)[\mu^{ext(\vec{y})}] = head(r')[\mu^{ext(\vec{y})}](\mathcal{C})$. Since, $dChase_{k+l}^{\text{csafe}}(QS_{\mathcal{C}})(\mathcal{C}) = dChase_k(QS_{\mathcal{C}})$, for any $j \leq l \leq j'$, and $dChase^{\text{csafe}}_{k+j'+1}(QS_{\mathcal{C}}) = dChase^{\text{csafe}}_{k+j'}(QS_{\mathcal{C}}) \cup head(r')[\mu^{ext(\vec{y})}]$, $dChase^{\text{csafe}}_{k+j'+1}(QS_{\mathcal{C}})(\mathcal{C}) = dChase_{k+1}(QS_{\mathcal{C}})$. Hence, the claim follows.

The following claim, which straightforwardly follows from claim 0, shows that, for csafe quad-systems its standard dChase is contained in its safe dChase.

Claim (1) Suppose **unCSafe** $\notin dChase^{\text{csafe}}(QS_{\mathcal{C}})$, then $dChase(QS_{\mathcal{C}}) \subseteq dChase^{\text{csafe}}(QS_{\mathcal{C}})$.

Claim below shows that the generation of originContext quads in csafe dChase is complete.

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

Since the only way a skolem blank node b gets generated in any iteration i of $dChase(QS_{\mathcal{C}})$ is by the application of a BR $r \in R$, i.e. when there $\exists r = body(r)(\vec{x}, \vec{z}) \rightarrow head(r)(\vec{x}, \vec{y}) \in R$, assignment μ, s.t. $applicable_R(r, \mu, dChase_{i-1}(QS_{\mathcal{C}}))$, and $b = y_j[\mu^{ext(\vec{y})}]$, for some $y_j \in \{\vec{y}\}$, and $dChase_i(QS_{\mathcal{C}}) = dChase_{i-1}(QS_{\mathcal{C}}) \cup head(r)(\vec{x}, \vec{y})[\mu^{ext(\vec{y})}]$. Also since $c \in originContexts(b)$, it should be the case that $c \in cScope(y_j, head(r))$. From claim 0, we know that there exists $j \geq 0$, s.t. $dChase_i(QS_{\mathcal{C}}) = dChase^{\text{csafe}}_{i+j}(QS_{\mathcal{C}})(\mathcal{C})$. W.l.o.g, assume that $i + j$ is the first such csafe dChase iteration. Hence, it follows that $applicable_{augC(R)}(r', \mu, dChase^{\text{csafe}}_{i+j-1}(QS_{\mathcal{C}}))$, where $r' = augC(r)$. Since, $head(r) \subseteq head(r')$, it should be the case that $c \in cScope(y_j, head(r'))$. Hence, by construction of $augC$, c_c: (y_j, originContext, c) $\in head(r')$, and as a result of application of μ on r' in iteration $i + j$, c_c: (b, originContext, c) gets generated in $dChase^{\text{csafe}}_{i+j}(QS_{\mathcal{C}})$. Hence, the claim holds.

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

- *sub-distance$(b, b') = 0$, if $b' = b$;*
- *sub-distance$(b, b') = \infty$, if $b \neq b'$ and b is not a descendant of b';*
- *sub-distance$(b, b') = min_{t \in \{\vec{x}[\mu]\}}\{$ sub-distance$(b, t)\} + 1$, if b' was generated by application of μ on $r = body(r)(\vec{x}, \vec{z}) \rightarrow head(r)(\vec{x}, \vec{y})$, i.e. $b' = y_j[\mu^{ext(\vec{y})}]$, for some $y_j \in \{\vec{y}\}$, and b is a descendant of b'.*

Claim (3) For any quad-system $QS_{\mathcal{C}} = \langle Q_{\mathcal{C}}, R\rangle$, if **unCSafe** $\notin dChase^{\text{csafe}}(QS_{\mathcal{C}})$, then for any two skolem blank nodes b, b' in $dChase(QS_{\mathcal{C}})$, if b is a descendant of b' then there exists a quad of the form c_c: $(b, \text{descendantOf}, b') \in dChase^{\text{csafe}}(QS_{\mathcal{C}})$.

Note by the definition of sub-distance that if b is a descendant of b', then sub-distance$(b, b') \in \mathbb{N}$. Assuming **unCSafe** $\notin dChase^{\text{csafe}}(QS_{\mathcal{C}})$, and b is a descendant of b', we approach the proof by induction on sub-distance(b, b').

base case Suppose sub-distance$(b, b') = 1$, then this implies that there exists $r = body(\vec{x}, \vec{z}) \rightarrow head(r)(\vec{x}, \vec{y})$, assignment μ s.t. b' was generated due to application of μ on r, i.e. $b' = y_j[\mu^{ext(\vec{y})}]$, for some $y_j \in \{\vec{y}\}$, and $b \in \{\vec{x}[\mu]\}$. This implies that there exists a dChase iteration i s.t. $applicable_R(r, \mu, dChase_i(QS_{\mathcal{C}}))$ and $dChase_{i+1}(QS_{\mathcal{C}}) = dChase_i(QS_{\mathcal{C}}) \cup apply(r, \mu)$. Since **unCSafe** $\notin dChase^{\text{csafe}}(QS_{\mathcal{C}})$, using claim 0, there exists $k \geq i$ s.t. $dChase_i(QS_{\mathcal{C}}) = dChase^{\text{csafe}}_k(QS_{\mathcal{C}})(\mathcal{C})$. W.l.o.g., let k be the first such csafe dChase iteration. This means that $applicable_{augC(R)}(r', \mu, dChase^{\text{csafe}}_k(QS_{\mathcal{C}}))$, where $r' = augC(r)$, and $dChase^{\text{csafe}}_{k+1} = dChase^{\text{csafe}}_k(QS_{\mathcal{C}}) \cup head(r')[\mu^{ext(\vec{y})}]$, and $b, b' \in head(r')[\mu^{ext(\vec{y})}]$, $b \in \{\vec{x}[\mu]\}$, $b' = y_j[\mu^{ext(\vec{y})}]$. By construction of $augC()$, since there exists a quad-pattern c_c: (x_l, descendantOf, y_j) $\in head(r')$, for any $x_l \in \{\vec{x}\}$, $y_j \in \{\vec{y}\}$, it follows that c_c: $(b, \text{descendantOf}, b') \in dChase^{\text{csafe}}_{k+1}(QS_{\mathcal{C}})$.

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

inductive step Suppose sub-distance$(b, b') = k + 1$, then there exists a $b'' \neq b$, assignment μ, and BR $r = body(r)(\vec{x}, \vec{z}) \rightarrow head(r)(\vec{x}, \vec{y}) \in R$ s.t. b' was generated due to the application of μ or r with $b'' \in \{\vec{x}[\mu]\}$, i.e. $b' = y_j[\mu^{ext(\vec{y})}]$, for $y_j \in \{\vec{y}\}$, and b is a descendant of b''. This implies that sub-distance$(b'', b') = 1$, and sub-distance$(b, b'') = k$, and hence by hypothesis c_c: (b, descendantOf, b'') $\in dChase^{\text{csafe}}(QS_{\mathcal{C}})$, and c_c: (b'', descendantOf, b') $\in dChase^{\text{csafe}}(QS_{\mathcal{C}})$. Hence, by construction of csafe dChase, c_c: (b, descendantOf, b') $\in dChase^{\text{csafe}}(QS_{\mathcal{C}})$.

Suppose $QS_\mathcal{C}$ is uncsafe, then by definition, there exists a blank nodes b, b' in $\mathbf{B}_{sk}(dChase(QS_\mathcal{C}))$, s.t. b is descendant of b', and $originContexts(b) = originContexts(b')$. By contradiction, if **unCSafe** $\notin dChase^{\text{csafe}}(QS_\mathcal{C})$, then by claim 1, $dChase(QS_\mathcal{C}) \subseteq dChase^{\text{csafe}}(QS_\mathcal{C})$. Since by claim 2, for any $c \in originContexts(b)$, there exists quads of the form c_c: (b, originContext, c) $\in dChase^{\text{csafe}}(QS_\mathcal{C})$ and for every $c' \in originContexts(b')$, there exists c_c: (b', originContext, c') $\in dChase^{\text{csafe}}(QS_\mathcal{C})$. Since $originContexts(b) = originContexts(b')$, it follows that $\{c \mid c_c$: (b, originContext, c) $\in dChase^{\text{csafe}}(QS_\mathcal{C})\} = \{c' \mid c_c$: ($b'$, originContext, c') $\in dChase^{\text{csafe}}(QS_\mathcal{C})\}$ Also by claim 3, since b is a descendant of b', there exists a quad of the form c_c: (b, descendantOf, b') in $dChase^{\text{csafe}}(QS_\mathcal{C})$. But, by construction of $dChase^{\text{csafe}}(QS_\mathcal{C})$, it should be the case that there exist a $b'' \in \mathbf{B}_{sk}(dChase^{\text{csafe}}(QS_\mathcal{C}))$, $r = body(r)(\vec{x}, \vec{z}) \to head(r)(\vec{x}, \vec{y}) \in augC(R)$, assignment μ s.t. b' was generated due to the application of μ on r, i.e. $b' = y_j[\mu^{ext(\vec{y})}]$ with $b'' \in \{\vec{x}[\mu]\}$, and c_c: (b, descendantOf, b'') $\in dChase^{\text{csafe}}(QS_\mathcal{C})$. But, since $\{c \mid c_c$: (b, originContext, c) $\in dChase^{\text{csafe}}(QS_\mathcal{C})\} = cScope(y_j, head(r_i))$, the method **unCSafeTest**(r, μ, $dChase_l^{\text{csafe}}(QS_\mathcal{C})$) should return True, for some $l \in \mathbb{N}$. Hence, it should be the case that **unCSafe** $\in dChase^{\text{csafe}}(QS_\mathcal{C})$, which is a contradiction to our assumption. Hence **unCSafe** $\in dChase^{\text{csafe}}(QS_\mathcal{C})$, if $dChase(QS_\mathcal{C})$ is uncsafe. $\square$