Nested Contextualised Views in the Web of Data

Anastasia Analyti[1] and Carlos V. Damásio[2] and Ioannis Pachoulakis[3]

[1] Institute of Computer Science, FORTH-ICS, Greece
[2] CENTRIA, Departamento de Informática Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa, Caparica, Portugal
[3] Dept. of Informatics Engineering, TEI of Crete, Greece
analyti@ics.forth.gr, cd@fct.unl.pt, ip@ie.teicrete.gr

Abstract. The notion of context is important since it can capture many of the interesting aspects of the way we understand the world such as relativity, locality, partiality, and context-dependence. In this work, we define the *contextual RDFS model theory*, called c-RDFS model theory for short, where a context is a triple of a name, a c-RDF graph, and a set of references (links) from resources to other contexts. This way chains of nested contextualized views are formed, where corresponding contexts can be complementary or conflicting. A collection of such chains forms a contextual structure. The c-RDFS model theory is developed extending the RDFS 1.1 model theory, which supports inferences both within contexts but also across contexts. Entailment of contextual structures is defined extending RDFS entailment between RDF graphs. For querying contextual structures, an extension of the SPARQL 1.0 language is proposed which allows querying one context at a time (thus delimiting a portion of interest) but also navigation through the references of a context reaching the contextual views of the desired resources.
Keywords: Contextualized views, contextual structures, RDFS 1.1 model theory, entailment, querying.

1 Introduction

During the recent years an increasing number of data providers adopted a set of best practices for publishing and connecting RDF data on the Web, leading to the creation of distributed dataspaces in the Web of Data [15]. Each RDF ontology is possibly associated with metadata information (like using the PROV-O ontology [19], currently a W3C recommendation), expressing, for example, its quality, certainty, authority, temporal and spatial status. Metadata can also relate RDF ontologies among themselves through their names [7, 6]. However, no notion has been presented for structuring the published RDF ontologies in a systematic way describing nested contextualized views. In other words, there is no extension of the RDFS theory that allows to declare that a context c is the view of a resource w.r.t. another context c'. In the Web of Data, resources are identified by their IRIs. However, the same IRI can have different meanings and associations in different contexts. Assuming that a RDF ontology corresponds to a context, merging different and non-compatible contexts will result in conflicting and meaningless information.

The motivation of this work is to extend the RDFS 1.1 model theory [14] so as to support contexts which are named RDF graphs [7, 6] containing IRIs contextualized within them and other foreign IRIs. Related contexts should be organized in contextual structures that link them based on their semantics. Thus, from one context we

can navigate to another context through an IRI that indicates its semantics. This way a chain of contexts can be formed and the sequence of IRIs that has to be followed to reach them indicates their semantics. Inference rules should be devised in order to migrate information from one context into another. Additionally, contextual structures should be able to be queried through an extension of the SPARQL 1.0 query language [25].

In this work, we define the *contextual RDFS model theory*, called c-RDFS model theory for short. In particular, we define a context c as a structure that (i) is associated with a name, nam_c, which is an IRI, (ii) contains a contextual RDF graph G_c, and (iii) contains a set of mappings (called *references*) from IRIs to other contexts. A *contextual* IRI has the form $[nam_c].u$, where c is a context and u is an IRI. In particular, $[nam_c].u$ represents a resource u within the view of context c. A *contextual RDF graph*, called c-RDF graph for short, is a set of *c-RDF triples* of the form (s, p, o), where the property p is an IRI or contextual IRI, and the subject s and object o are blank nodes, IRIs, literals, or contextual IRIs. A non-contextual IRI within a context c denotes a resource local in c. That is a contextual IRI $[nam_c].u$ and the non-contextual IRI u within a context c are representing the same resource. On the other hand, it possible to have a contextual IRI $[nam_{c'}].u$ within a context c, indicating the resource u as viewed from another context c'.

Obviously, every RDF graph is a c-RDF graph. We indicate that the reference of an IRI u from the point of view of a context c is another context c' by writing $ref_c(u) = nam_{c'}$. The context c' provides details of resource u within context c. This way a chain of nested contexts can be formed indicating the semantics of the initial context. For example, the notion of city in the 15th century may be represented by a context describing geography concepts, connected by a reference to a context describing 15th century, which contains the concept of city. This simple example is indicated in Figure 1[4], where we see three contexts, named `Geography`[5], `c`, and `c'`. In particular, the context named `c` is the reference of the IRI `15th century` within the context named `Geography`. Similarly, the context named `c'` is the reference of the IRI `20th century` within the context named `Geography`. As it is indicated in the figure, in the 15th century, a city is a settlement of more than 5000 people, while in the 20th century, a city is a settlement of more than 10000 people. Additionally, the city *Chandax* was later renamed *Heraklion* and B in the 15th century was a village while in the 20th century became a city. Further, in the figure, we see that in the 15th century *Chandax* has a fortress while *Heraklion* has an airport. Note that the non-contextual IRI `Chandax` in the context named `Geography` is considered identical to the contextual IRI [`Geography`].`Chandax` in the contexts named `c` and `c'`. Further note that due to the fact that [`Geography`].`B` was village in the context named `c` and it became a city in the context named `c'`, the contexts named `c` and `c'` are essentially conflicting.

A *contextual structure* CS is a set of contexts satisfying the criteria: (i) if $c \in CS$ then all of the subcontexts of c should belong to CS and (ii) if $[nam_c].u$ appears in a context of CS then c should belong to CS. For example, each set of contexts within figures 1 and 2 forms a contextual structure.

This work was inspired by the work in [3]. However, while the authors of [3] develop their own model theory, basically based on the TELOS conceptual model [23],

[4] Note that arrows with the label `ref` indicate references. Bold arrows indicate `subClassOf` relationships and dotted arrows indicate `type` relationships.

[5] In this paper, for simplification of presentation, we ignore namespaces from the IRIs.

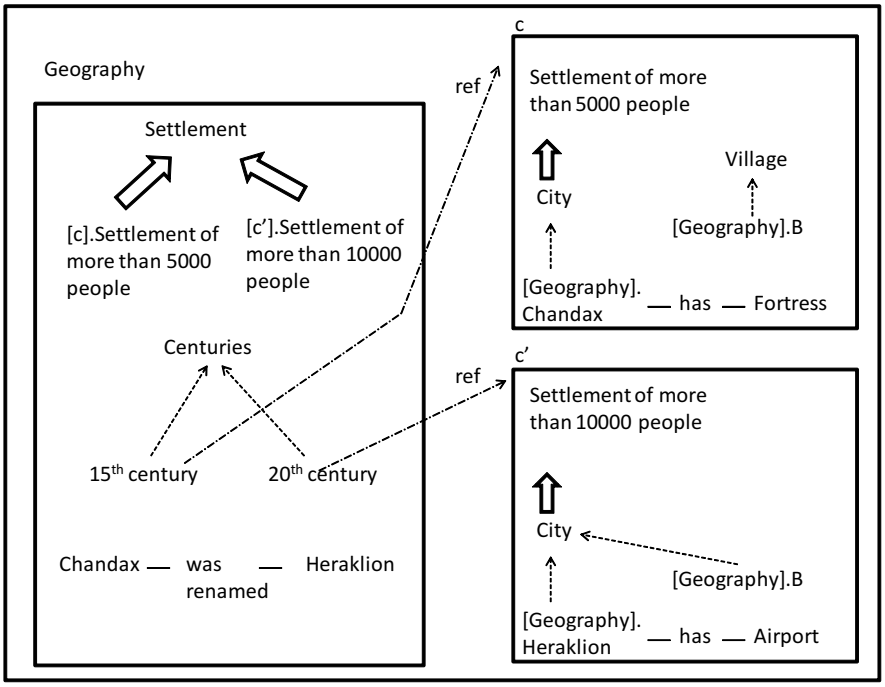

Fig. 1. A contextual structure about geography

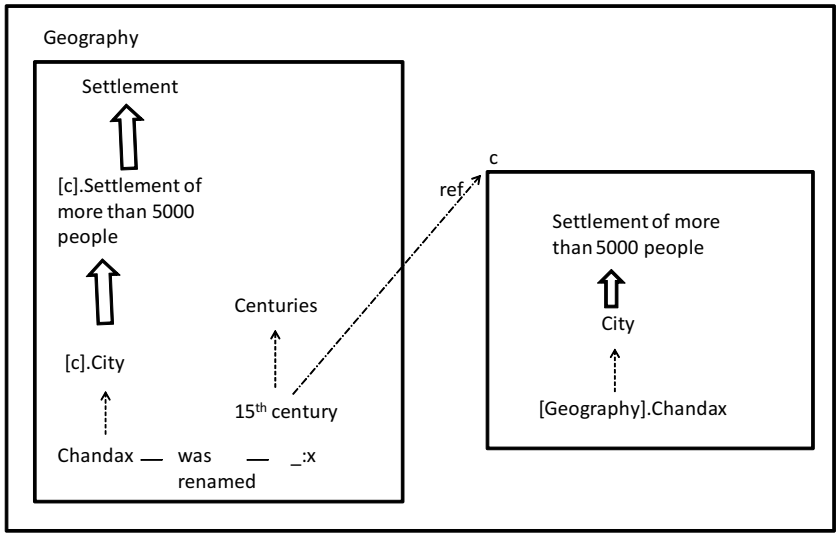

Fig. 2. An entailed contextual structure about geography

we extend the RDFS 1.1 model theory [14]. All inferences made based on the RDFS model theory are also supported by our model theory within a context. However, additional rules have been added for spanning contexts within a contextual structure. Based on these rules we define (c-*RDFS*) *entailment* between contextual structures. For example, the contextual structure CS in Figure 1 entails the contextual structure CS' in Figure 2. Note that the contexts named `Geography` and `c` of contextual structure CS' have been mapped to the contexts named `Geography` and `c` of contextual structure CS, while the contexts named `Geography` and c of CS' contain a subset of the inferences made by our c-RDFS model theory. Note that while contexts names are unique within a contextual structure, different contextual structures may contain different contexts with the same name.

In particular, note that in Figure 2, the context named `Geography` contains the c-RDF triple (`Chandax`, `type`, [`c`].`City`) because in Figure 1, the context named `c` contains the c-RDF triples ([`Geography`].`Chandax`, `type`, `City`) and (`City`, `subClassOf`, `Settlement of more than 5000 people`) and the context named `Geography` contains the c-RDF triple (`Settlement`, `subClassOf`, [`c`].`Settlement of more than 5000 people`). Due to the latter c-RDF triple, a transfer of knowledge is happening from context the context named `c` to the context named `Geography` of Figure 1.
In particular, in this paper:

- We define contexts and contextual structures, showing that they support nested contextualized views in the Web of Data. Additionally, we show that sharing of views is also supported by our model. Thus, the same context can be reached through different paths.
- We define c-RDFS interpretations of a contextual structure extending RDFS interpretations. Then, based on this, we define satisfaction of a contextual structure by a c-RDFS interpretation. We also define (c-RDFS) entailment between contextual structures extending the RDFS entailment between RDF graphs.
- We provide a set of inference rules, extending the RDFS inference rules, whose closure defines the largest contextual structure $cl(CS)$, called *closure* of CS, entailed by a contextual structure CS. It holds that every contextual structure entailed by $cl(CS)$ is also entailed by CS and vise-versa. Based on the notion of the closure of a contextual structure, we provide a procedure for checking entailment between contextual structures, whose time-complexity is not higher than checking RDFS entailment between RDF graphs, that is NP-complete.
- We extend the SPARQL 1.0 query language on RDF graphs [25], defining the c-SPARQL query language operating on contextual structures such that queries on contexts and contextual IRIs are supported. Using c-SPARQL, navigation through the references of a context c is possible, reaching any subcontext of c.

The rest of the paper is organized as follows: In Section 2, we define contexts and contextual structures. and provide examples. In Section 3, we define c-RDFS interpretations of a contextual structure, the models of a contextual structure, and entailment between contextual structures. In Section 4, we provide a set of inference rules which define the closure of a contextual structure and we provide a method for checking entailment between contextual structures. In Section 5, we define the incorporation of a context into another context which causes also merging of their corresponding subcontexts. In Section 6, we present the c-SPARQL query language for querying contextual structures. In Section 7, we describe related work. Finally, Section 8 contains concluding remarks and directions for further research. Appendix

A contains a list of symbols and Appendix B provides the proof of the propositions appearing in the main paper. Due to space limitations, Appendix B is provided online in the web address `http://www.ics.forth.gr/~analyti/Papers_2/RDF_context_journal_appendix.pdf`.

2 Contextual Structures

In this Section, we define contexts, contextual structures, and provide examples.

A *name* is an IRI reference or a literal [14]. A (Web) *vocabulary* V is a set of names. We denote the set of names by Nam, the set of all IRI references by IRI, and the set of all literals by $\mathcal{LIT}$. We consider a set BN of blank nodes, such that the sets BN, IRI, $\mathcal{LIT}$ are pairwise disjoint. The vocabulary of RDF, $\mathcal{V}_{RDF}$, is a set of IRI references in the *rdf*: namespace [14], and the vocabulary of RDFS, $\mathcal{V}_{RDFS}$, is a set of IRI references in the *rdfs*: namespace [14]. Let $D = \{rdf{:}langString, xsd{:}string\}$, which is the set of datatypes supported by the RDF 1.1 semantics. Let $CNAM$ be the set of contexts names, that is IRIs which include a special IRI denoted by nil. The special context with name nil holds all names with a global meaning, that is these in $\mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{LIT} \cup D$.

First, we define a *contextual name* as a term $[u].w$, where $u \in CNAM$ and w is a name, and it represents the name w within the context having name u. If $w \in \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{LIT} \cup D$ then the only contextual names allowed are these of the form $[nil].w$. We denote the set of contextual names by c-Nam. A c-*vocabulary* V is a set of contextual names. We denote the set of contextual IRIs by c-IRI and the set of contextual literals by c-$\mathcal{LIT}$. For uniformity reasons, blank nodes can be also contextualized by the special context named nil. We denote contextual blank nodes by c-BN.

Now, we define c-RDF triples extending RDF triples.

Definition 1 (c-RDF triple). A c-*RDF triple* G has the form (s, p, o), where $p \in IRI \cup$ c-IRI, and $s, o \in Nam \cup$ c-$Nam \cup BN \cup$ c-BN. The symbol s is called the *subject* of the triple, the symbol p is called the *property* of the triple, and the symbol o is called the *object* of the triple. □

Our choice of allowing literals appearing in the subject position is based on our intuition that this case can naturally appear in knowledge representation. SPARQL [25] and de Bruijn et al. [8] also consider literals in the subject position of RDF triples. Note that the RDFS model theory [14] does not allow literals to appear in the subject position of an RDF triple.

Based on the notion of a c-RDF triple, we define a c-RDF graph extending RDF graphs.

Definition 2 (c-RDF graph). A *contextual RDF graph*, called c-*RDF graph* for short, G is a set of c-RDF triples. □

As it is obvious, a c-RDF graph can contain properties connecting possibly contextual resources. For example, a c-RDF graph may contain the c-RDF triple $([\texttt{Geography}].\texttt{Herakleio}, \texttt{type}, \texttt{city})$, indicating that the geographical concept `Herakleio` is a city.

A c-RDF triple is said *complete* if it contains only contextual names and contextual blank nodes. A c-RDF graph is said *complete* if it contains only complete c-RDF triples.

Below, we formally define a context as a triple of its name, its c-RDF graph, and a partial function that maps IRIs to other contexts.

Definition 3. A *context* c is a structure $c = \langle nam_c, G_c, ref_c \rangle$, where $nam_c \in IRI$ is the name of c, G_c is a c-RDF graph, and ref_c is a partial mapping from IRIs to $CNAM - \{nil\}$. □

The *vocabulary of* c, denoted by V_c, is the union of (i) the contextual names appearing in G_c, (ii) $\{[nam_c].w \mid w \notin \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup D$ is a non-contextual IRI appearing in $G_c\}$, (ii) $\{[nil].w \mid w \in \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{LIT} \cup BN \cup D$ appears non-contextualized in $G_c\}$.

We assume a special context $NIL = \langle nil, \{\}, m \rangle$, where m is the empty mapping. We denote the set of contexts by CXT.

Note that in the vocabulary of c, V_c, non-contextual IRIs appearing in c and that do not belong to $\mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup D$ are becoming contextualized within c, while non-contextual names in $\mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{LIT} \cup D$ appearing in G_c are becoming contextualized within the special context *NIL*. The latter is because names in the vocabularies of RDF and RDFS, as well as literals and datatypes in D, are assumed to have a global meaning.

Note that the partial mapping ref_c maps IRIs to names of contexts. If $ref_c(u) = u'$ then we say that the reference (partial view) of resource u withing context c is the context with name u'. Of course, within the context c, a resource may not have a view. Additionally, since u' is the name of a context c', c' contains its own IRIs which may have references. That is, a contextual structure is built through chains of contexts which should be acyclic and finite. If a context c is reached through a chain of references starting from a context c' then c is called *subcontext* of c'. Obviously, it is not reasonable for the view of a resource within a subcontext of a context c to be the context c itself. That is, at the end of a contextual structure, we reach contexts which have no references and these are called *leaf contexts*.

Definition 4. A *contextual structure* is a set of contexts CS which does not include *NIL* such that (i) if $c \in CS$ then all of the subcontexts of c belong to CS, (ii) if $[u].w$ appears in a context $c \in CS$, where $u \neq nil$ then there is a context $c' \in CS$ with the name u, (iii) context names are unique within CS, and (iv) each chain of subcontexts of a context $c \in CS$ reaches a leaf context. □

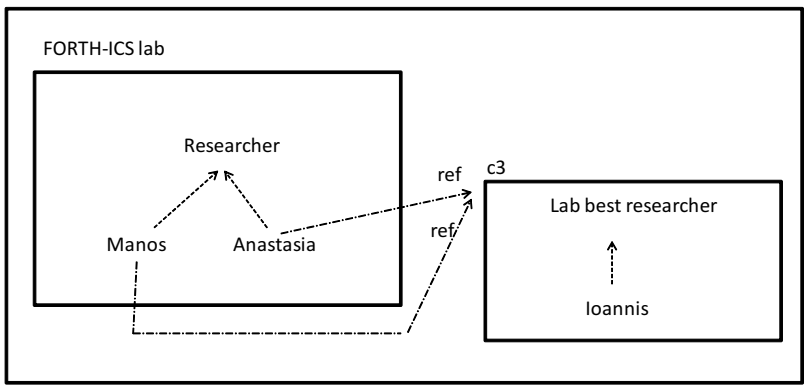

Fig. 3. A contextual structure showing shared views

Of course within a contextual structure, contexts can be reached through several paths indicating sharing of views, as shown in Figure 3. In the figure, we see two FORTH-ICS lab researcher having the same opinion about the lab.

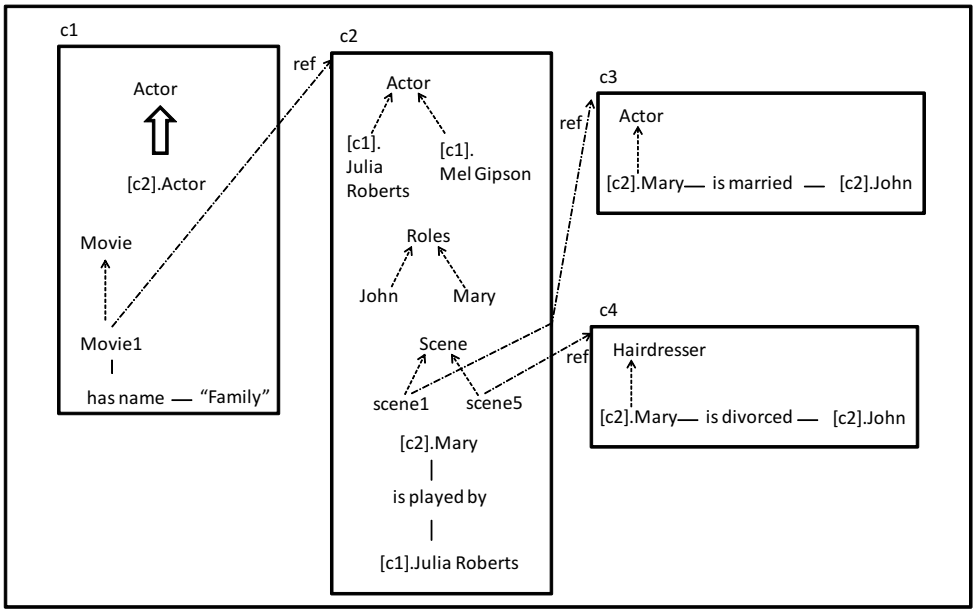

Fig. 4. A contextual structure about movies

Example 1. We now describe an example on a contextual structure about movies shown in Figure 4. In the context named `c1`, it is shown a class `Movie` with an instance a particular movie `Movie1`, called *Family*. `Movie1` refers (within the context named `c1`) to a context named `c2`, providing details about `Movie1`. Within the context named `c2` is provided information about the actors of the movie, the roles, and that the role Mary is played by Julia Roberts. In the same context, the class `Scene` has instances `scene1` and `scene5`, each referring to a different context providing details about the scene. In particular, `scene1` refers to context named `c3`, which describe that Mary is married to John and that the job of Mary is actress. Similarly, `scene5` refers to context named `c4`, which describe that Mary is divorced from John and the the new job of Mary is hairdresser. Note that in the context named `c1`, there is a `subClassOf` relationship from the contextual IRI [`c2`].`Actor` to the non-contextual IRI `Actor`. As we will see, based on this relationship, the actors of the context named `c2` are becoming actors in the context named `c1`.

Additionally, note that the semantics of context `c3` is provided by the sequence of IRIS `Movie1.scene1` and the semantics of context `c4` is provided by the sequence of IRIS `Movie1.scene5`. □

Note that based on our method, alternative representations of resources are supported both within contexts but also through their references. For example, the resource 15^{th}`century` will have different associations within a context describing the geography of Greece and within a context describing the geography of Italy. Also, its references with respect to these contexts will be different.

Another prominent feature is that our theory allows users to focus on a specific context at a time and then navigate through the references of the resources inside the contextual structure, as desired. Thus, the user does not get lost in a large RDF graph, providing all information in a unique context.

3 Models of Conceptual Structures and Entailment

In this Section, we define c-simple interpretations, c-RDFS interpretations, the models of a contextual structure and entailment between two contextual structures.

Below we define a c-simple interpretation of a c-vocabulary extending the definition of a pre-interpretation of a vocabulary in the RDFS model theory [14].

Definition 5 (c-Simple interpretation of a c-vocabulary). A *c-interpretation* I of a c-vocabulary V_I consists of:

- A non-empty set of resources Res_I of the form $[u].r$, called the *domain* or *universe* of I.
- A set of properties $Prop_I$ of the form $[u].r$.
- A vocabulary interpretation mapping $I_{\mathtt{V}} : V_I \cap \mathtt{c}\text{-}IRI \rightarrow Res_I \cup Prop_I$ such that for each $[u].u' \in V_I \cap \mathtt{c}\text{-}IRI$, there is r where $I_{\mathtt{V}}([u].u') = [u].r$.
- A property extension mapping $PT_I : Prop_I \rightarrow \mathcal{P}(Res_I \times Res_I)$[6].
- A partial mapping $IL_I : V_I \cap \mathtt{c}\text{-}\mathcal{LIT} \rightarrow Res_I$ such that for $[nil].l \in V_I \cap \mathtt{c}\text{-}\mathcal{LIT}$, there is r where $IL_I([nil].l) = [nil].r$.
- A partial mapping ref_I from IRIs to $CNAM \setminus \{nil\}$.

We define the partial mapping: $I : V_I \rightarrow Res_I \cup Prop_I$ such that: (i) $I(x) = I_{\mathtt{V}}(x)$, $\forall x \in V_I \cap \mathtt{c}\text{-}IRI$, and (ii) $I(x) = IL_I(x),\ \forall x \in V_I \cap \mathtt{c}\text{-}\mathcal{LIT}$. □

As we will see latter the partial mapping ref_I from IRIs to $CNAM \setminus \{nil\}$ is needed for supporting references of contexts.

Let CS be a contextual structure. We define by n_{CS} to be the maximum of 1 and the largest i such that $rdf{:}\_i$ appears in a context of CS. Recall that the $rdf{:}\_i$ properties are used in RDFS [14] to express members of containers (i.e. bags, sequences, and alternatives), which are in practice finitely limited. We define $\mathcal{V}_{RDF}^{\#n} = \mathcal{V}_{RDF} - \{rdf{:}\_i \mid i > n\}$.

Definition 6 (Vocabulary of a contextual structure). We define the *vocabulary of a contextual structure* CS as follows:

$$V_{CS} = \cup\{V_c \mid c \in CS\} \bigcup \{[nil].u \mid u \in \mathcal{V}_{RDF}^{\#n_{CS}} \cup \mathcal{V}_{RDFS} \cup D\}$$

□

Below, we extend the definition of the composition of a simple interpretation of a vocabulary V and a valuation [14].

Definition 7 (Composition of a c-simple interpretation and a valuation). Let I be a c-simple interpretation of a c-vocabulary V and v be a partial mapping from $\mathtt{c}\text{-}BN$ to a set Res (called *valuation*). We define (i) $[I{+}v](x) = v(x)$, if $x \in \mathtt{c}\text{-}BN$ and (ii) $[I + v](x) = I(x)$, if $x \in V$. □

[6] The symbol $\mathcal{P}(S)$ denotes the powerset of set S.

Below we define satisfaction of a context by a c-simple interpretation and a valuation by extending satisfaction of an RDF graph and a valuation [14]. First, we give an auxiliary definition.

Definition 8 (completion of a c-RDF graph). Let c be a context. We denote by $compl\_graph(c)$ the complete c-RDF graph that is derived if we replace in G_c (i) all non-contextual names and blank nodes w in $\mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{LIT} \cup BN \cup D$ appearing in G_c by $[nil].w$, (ii) the rest non-contextual IRIs u by $[nam_c].u$. □

Obviously, the vocabulary of a context c is the set of contextual names appearing in $compl\_graph(c)$.

Definition 9 (Satisfaction of a c-RDF triple and a context by a c-simple interpretation and a valuation). Let I be a c-simple interpretation of a c-vocabulary V, let c be a context, and let v be a valuation.

- If t is a c-RDF triple $([u].s, [u'].p, [u''].o)$ then $I, v \models t$ if $\langle [I + v]([u].s), [I + v]([u''].o) \rangle \in PT_I(I([u'].p))\rangle$.
- $I, v \models c$, if (i) $I, v \models t$, for every $t \in compl\_graph(c)$ and (ii) if $ref_c(u) = u'$ then $ref_I(u) = u'$. □

Let t is a c-RDF triple. We define $I \models t$, if there is a valuation v such that $I, v \models t$. Let c be a context. We define $I \models c$, if there is a valuation v such that $I, v \models c$.

Below we define a c-RDFS interpretation of a contextual structure CS extending the definition of an RDFS interpretation [14].

Definition 10 (c-RDFS interpretation of a contextual structure). A c-RDFS interpretation of a contextual structure CS is a set $\mathsf{I} = \{I_c \mid c \in CS\}$ such that each I_c is a c-simple interpretation of a c-vocabulary $V_{I_c} \supseteq V_{CS}$, extended with the set of classes Cls_{I_c} and the class extension mapping $CT_{I_c} : Cls_{I_c} \to \mathcal{P}(Res_{I_c})$.

It should be enforced that, if $c, c' \in CS$ and $[u].w \in V_{I_c} \cap V_{I_{c'}}$ then $I_c([u].w) = I_{c'}([u].w)$.
Additionally, the following c-*RDFS interpretation conditions* should be obeyed:[7]

1. $x \in CT_{I_c}(y)$ iff $\langle x, y \rangle \in PT_{I_c}(I_c(\texttt{type}))$,
2. The ontological categories are defined as follows:
   $Prop_{I_c} = CT_{I_c}(I_c(\texttt{Property}))$ $Cls_{I_c} = CT_{I_c}(I_c(\texttt{Class}))$
   $Res_{I_c} = CT_{I_c}(I_c(\texttt{Resource}))$
3. If $\langle x, y \rangle \in PT_{I_c}(I_c(\texttt{domain}))$ and $\langle z, w \rangle \in PT_{I_c}(x)$ then $z \in CT_{I_c}(y)$.
4. If $\langle x, y \rangle \in PT_{I_c}(I_c(\texttt{range}))$ and $\langle z, w \rangle \in PT_{I_c}(x)$ then $w \in CT_{I_c}(y)$.
5. If $x \in Cls_{I_c}$ then $\langle x, I_c(\texttt{Resource}) \rangle \in PT_{I_c}(I_c(\texttt{subClassOf}))$.
6. If $\langle x, y \rangle \in PT_{I_c}(I_c(\texttt{subClassOf}))$ then $x, y \in Cls_{I_c}$ and $CT_{I_c}(x) \subseteq CT_{I_c}(y)$.
7. If $c' \in CS$ and $\langle [nam_{c'}].x, y \rangle \in PT_{I_c}(I_c(\texttt{subClassOf}))$ then $CT_{I_{c'}}([nam_{c'}].x) \subseteq CT_{I_c}(y)$.

[7] To simplify notation, we have replaced $[nil].x$, where $x \in \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup D$, by x without the corresponding namespace for $x \in \mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS}$.

8. $PT_{I_c}(I_c(\texttt{subClassOf}))$ is a reflexive and transitive relation on Cls_{I_c}.
9. If $c' \in CS$, $\langle [nam_{c'}].x, y\rangle \in PT_{I_c}(I_c(\texttt{subClassOf}))$ and $\langle z, [nam_{c'}].x\rangle \in PT_{I_{c'}}(I_{c'}(\texttt{subClassOf}))$ then $\langle z, y\rangle \in PT_{I_c}(I_c(\texttt{subClassOf}))$.

10. If $\langle x, y\rangle \in PT_{I_c}(I_c(\texttt{subPropertyOf}))$ then $x, y \in Prop_{I_c}$ and $PT_{I_c}(x) \subseteq PT_{I_c}(y)$.
11. If $c' \in CS$ and $\langle [nam_{c'}].x, y\rangle \in PT_{I_c}(I_c(\texttt{subPropertyOf}))$ then $PT_{I_{c'}}([nam_{c'}].x) \subseteq PT_{I_c}(y)$.

12. $PT_{I_c}(I_c(\texttt{subPropertyOf}))$ is a reflexive and transitive relation on $Prop_{I_c}$.
13. If $c' \in CS$, $\langle [nam_{c'}].x, y\rangle \in PT_{I_c}(I_c(\texttt{subPropertyOf}))$, and $\langle z, [nam_{c'}].x\rangle \in PT_{I_{c'}}(I_{c'}(\texttt{subPropertyOf}))$ then $\langle z, y\rangle \in PT_{I_c}(I_c(\texttt{subPropertyOf}))$.

14. If $x \in CT_{I_c}(I_c(\texttt{Datatype}))$ then $\langle x, I_c(\texttt{Literal})\rangle \in PT_{I_c}(I_c(\texttt{subClassOf}))$.
15. If $x \in CT_{I_c}(I_c(\texttt{ContainerMembershipProperty}))$ then $\langle x, I_c(\texttt{member})\rangle \in PT_{I_c}(I_c(\texttt{subPropertyOf}))$.

16. If $d \in D$ then $I_c([nil].d)$ is the datatype identified by d and $I_c([nil].d) \in CT_{I_c}(I_c(\texttt{Datatype}))$.

17. For every contextual language-tagged string $[nil].E \in V_{I_c}$ with lexical form s and language tag t, it holds that $I_c([nil].E) = [nil].\langle s, t'\rangle$, where t' is t converted to lower case using US-ASCII rules.
18. For every contextual well-typed string $[nil].\text{“}s\text{”}\hat{}\hat{}xsd{:}string \in V_{I_c}$, it holds that $I_c([nil].\text{“}s\text{”}\hat{}\hat{}xsd{:}string) = [nil].\text{“}s\text{”}$. Additionally, for every contextual not well-typed string $[nil].\text{“}s\text{”}\hat{}\hat{}xsd{:}string \in V_{I_c}$, it holds that $I_c([nil].\text{“}s\text{”}\hat{}\hat{}xsd{:}string)$ is undefined.

19. $CT_{I_c}(I_c(\texttt{rdf:langString})) = \{I_c([nil].E) \mid [nil].E$ is a contextual language-tagged string appearing in $V_{I_c}\}$.
20. $CT_{I_c}(I_c(\texttt{xsd:string})) = \{[nil].\text{“}s\text{”} \mid [nil].\text{“}s\text{”}\hat{}\hat{}xsd{:}string$ is a contextual well-typed string appearing in $V_{I_c}\}$.

21. if (s, p, o) is an RDF or RDFS axiomatic triple [14], that contain IRI references in $\mathcal{V}_{RDF}^{\#n_{CS}} \cup \mathcal{V}_{RDFS}$ then I_c satisfies the c-triple $([nil].s, [nil].p, [nil].o)$. □

In the previous definition, the only not familiar conditions from the RDFS model theory [14] are conditions 7, 9, 11, and 13. Condition 7 expresses that if a class $[nam_{c'}].x$ is subclass of class y in context c then every resource classified to $[nam_{c'}].x$ in context c' is also classified to y in context c. Condition 9 expresses that if a class $[nam_{c'}].x$ is subclass of class y in context c and a class z is subclass of class $[nam_{c'}].x$ in context c' then class z is subclass of class y in context c. Condition 11 expresses that if a property $[nam_{c'}].x$ is subproperty of property y in context c then every two resources related by $[nam_{c'}].x$ in context c' are also related by y in context c. Condition 13 expresses that if a property $[nam_{c'}].x$ is subproperty of a property y in context c and a property z is subproperty of property $[nam_{c'}].x$ in context c' then property z is subproperty of property y in context c. Based on the previously

described conditions a transfer of knowledge is happening from context c' to context c.

Thus, any transfer of knowledge between contexts involves `subClassOf` and `subPropertyOf` relationships. In particular, let $c, c' \in CS$. The c-RDF triples $([nam_{c'}].w, \texttt{subClassOf}, o)$ and $([nam_{c'}].w, \texttt{subPropertyOf}, o)$ inside in G_c imply that transfer of knowledge from c' to c is desired.

The following definition extends satisfaction of RDF graphs by a simple interpretation [14].

Definition 11 (satisfaction of a contextual structure by a c-RDFS interpretation). Let CS be a conceptual structure, let $\mathsf{I} = \{I_c \mid c \in CS\}$ be a c-RDFS interpretation of CS, and let v be a valuation. We define:

- $\mathsf{I}, v \models CS$ if, for all $c \in CS$, $I_c, v \models c$.
- I *satisfies* CS, denoted by $\mathsf{I} \models CS$, if there is valuation v such that, for all $c \in CS$, $I_c, v \models c$.

If $\mathsf{I} \models CS$ then I is called *model* of CS. The set of models of a contextual structure CS is denoted by $\mathcal{M}(CS)$. □

We now define entailment between contextual structures showing that it extends RDFS entailment between two RDF graphs [14].

First, we define the auxiliary definition $nam_{CS} = \{nam_c \mid c \in CS\}$.

Definition 12 (Entailment between contextual structures). A contextual structure CS (c-*RDFS*) *entails* a contextual structure CS', denoted by $CS \models CS'$, if (i) $nam_{CS'} \subseteq nam_{CS}$ and (ii) for each c-RDFS interpretation $\mathsf{I} = \{I_c \mid c \in CS\}$ that satisfies CS, it holds that $\mathsf{I}' = \{I'_c \mid c \in CS', I'_c = I_{c'}, c' \in CS, nam_c = nam_{c'}\}$ is a c-RDFS interpretation that satisfies CS'. □

Example 2. As we shall see in Example 4, after providing a method for checking entailment between contextual structures, the contextual structure CS in Figure 1 entails the contextual structure CS' in Figure 2. □

The following proposition shows that entailment between contextual structures extends RDFS entailment between two RDF graphs.

Proposition 1. Let G, G' be RDF graphs such that $max(\{i \in \mathbb{N} \mid \texttt{rdf:\_i}$ appears in $G'\}) \leq max(\{i \in \mathbb{N} \mid \texttt{rdf:\_i}$ appears in $G)\}$. Consider now the contextual structures $CS = \{\langle dummy, G, m\rangle\}$ and $CS' = \{\langle dummy, G', m\rangle\}$, where m is the empty mapping. Then, $G \models^{RDFS} G'$ iff $CS \models CS'$. □

4 Closure of a Contextual Structure

We now define the *closure* of a contextual structure CS, denoted by $cl(CS)$, such that $cl(CS) = \{cl(c, CS) \mid c \in CS\}$, where $cl(c, CS)$ is a new context with the same name as c, which will be defined in this Section. The closure of a contextual structure CS is the largest contextual structure entailed by CS in the sense that if we add a new c-RDF triple that contains only contextual names to a context $cl(c, CS)$, the extended contextual structure will no longer be entailed by CS. Based on $cl(CS)$, we are able to define a procedure for verifying the entailments of CS.

To generate $cl(CS)$, we will define a definite logic program P_{CS} which uses a single 7-ary predicate $H(nam_c, u_s, s, u_p, p, u_o, o)$ indicating that the complete c-RDF triple $([u_s].s, [u_p].p, [u_o].o)$ holds in $cl(c, CS)$.

We denote by P_c, where $c \in CS$, the following set of rules:

c-Simple Interpretation Rules

(1) $H(nam_c, ?u_z, ?z, nil, \texttt{type}, nil, \texttt{Property}) \leftarrow H(nam_c, ?u_x, ?x, ?u_z, ?z, ?u_y, ?y).$

(2) $H(nam_c, ?u_x, ?x, nil, \texttt{type}, nil, \texttt{Resource}) \leftarrow H(nam_c, ?u_x, ?x, ?u_z, ?z, ?u_y, ?y).$

(3) $H(nam_c, ?u_y, ?y, nil, \texttt{type}, nil, \texttt{Resource}) \leftarrow H(nam_c, ?u_x, ?x, ?u_z, ?z, ?u_y, ?y).$

c-RDFS Interpretation Rules

(1) $H(nam_c, ?u_z, ?z, nil, \texttt{type}, ?u_y, ?y) \leftarrow H(nam_c, ?u_x, ?x, nil, \texttt{domain}, ?u_y, ?y), H(nam_c, ?u_z, ?z, ?u_x, ?x, ?u_w, ?w).$

(2) $H(nam_c, ?u_w, ?w, nil, \texttt{type}, ?nam_y, ?y) \leftarrow H(nam_c, ?u_x, ?x, nil, \texttt{range}, ?u_y, ?y), H(nam_c, ?u_z, ?z, ?u_x, ?x, ?u_w, ?w).$

(3) $H(nam_c, ?u_x, ?x, nil, \texttt{subClassOf}, nil, \texttt{Resource}) \leftarrow H(nam_c, ?u_x, ?x, nil, \texttt{type}, nil, \texttt{Class}).$

(4) $H(nam_c, ?u_z, ?z, nil, \texttt{type}, ?u_y, ?y) \leftarrow H(nam_c, ?u_x, ?x, nil, \texttt{subClassOf}, ?u_y, ?y), H(nam_c, ?u_z, ?z, , nil, \texttt{type}, ?u_x, ?x).$

Traversing contexts rule (1)

(5) $H(nam_c, ?u_z, ?z, nil, \texttt{type}, ?u_y, ?y) \leftarrow H(nam_c, ?u_x, ?x, nil, \texttt{subClassOf}, ?u_y, ?y), H(?nam_x, ?u_z, ?z, , nil, \texttt{type}, ?u_x, ?x).$

(6) $H(nam_c, ?u_x, ?x, nil, \texttt{subClassOf}, ?u_x, ?x) \leftarrow H(nam_c, ?u_x, ?x, nil, \texttt{type}, nil, \texttt{Class}).$

(7) $H(nam_c, ?u_z, ?z, nil, \texttt{subClassOf}, ?u_y, ?y) \leftarrow H(nam_c, ?u_x, ?x, nil, \texttt{subClassOf}, ?y_y, ?y), H(nam_c, ?u_z, ?z, nil, \texttt{subClassOf}, ?u_x, ?x).$

Traversing contexts rule (2)

(8) $H(nam_c, ?u_z, ?z, nil, \texttt{subClassOf}, ?u_y, ?y) \leftarrow H(nam_c, ?u_x, ?x, nil, \texttt{subClassOf}, ?u_y, ?y), H(?nam_x, ?u_z, ?z, nil, \texttt{subClassOf}, ?u_x, ?x).$

(9) $H(nam_c, ?u_{z_1}, ?z_1, ?u_y, ?y, ?u_{z_2}, ?z_2) \leftarrow H(nam_c, ?u_x, ?x, nil, \texttt{subPropertyOf}, ?u_y, ?y), H(nam_c, ?u_{z_1}, ?z_1, ?u_x, ?x, ?u_{z_2}, ?z_2).$

Traversing contexts rule (3)

(10) $H(nam_c, ?u_{z_1}, ?z_1, ?u_y, ?y, ?u_{z_2}, ?z_2) \leftarrow H(nam_c, ?u_x, ?x, nil, \texttt{subPropertyOf}, ?u_y, ?y), H(?nam_x, ?u_{z_1}, ?z_1, ?u_x, ?x, ?u_{z_2}, ?z_2).$

(11) $H(nam_c, ?u_x, ?x, nil, \texttt{subPropertyOf}, ?u_x, ?x) \leftarrow H(nam_c, ?u_x, ?x, nil, \texttt{type}, nil, \texttt{Property})$.
(12) $H(nam_c, ?u_z, ?z, nil, \texttt{subPropertyOf}, ?u_y, ?y) \leftarrow H(nam_c, ?u_x, ?x, nil, \texttt{subPropertyOf}, ?u_y, ?y), H(nam_c, ?u_z, ?z, nil, \texttt{subPropertyOf}, ?u_x, ?x)$.

Traversing contexts rule (4)
(13) $H(nam_c, ?u_z, ?z, nil, \texttt{subPropertyOf}, ?u_y, ?y) \leftarrow H(nam_c, ?u_x, ?x, nil, \texttt{subPropertyOf}, ?u_y, ?y), H(?nam_z, ?u_z, ?z, nil, \texttt{subPropertyOf}, ?u_x, ?x)$.

(14) $H(nam_c, ?u_x, ?x, nil, \texttt{subClassOf}, nil, \texttt{Literal}) \leftarrow H(nam_c, ?u_x, ?x, nil, \texttt{type}, nil, \texttt{Datatype})$.
(15) $H(nam_c, ?u_x, ?x, nil, \texttt{subPropertyOf}, nil, \texttt{member}) \leftarrow H(nam_c, ?u_x, ?x, nil, \texttt{type}, nil, \texttt{ContainerMembershipProperty})$.

For $d \in D$:
(16) $H(nam_c, nil, d, nil, \texttt{type}, nil, \texttt{Datatype}) \leftarrow \texttt{true}$.

For each $[nil].\text{“}s\text{”}\hat{}\hat{}d \in V_{CS}$, where $d \in D$:
(17) $H(nam_c, nil, \text{“}s\text{”}\hat{}\hat{}d, nil, \texttt{type}, nil, d) \leftarrow \texttt{true}$.

For each $[nil].E \in V_{CS}$, where E is not a language-tagged string:
(18) $\texttt{false} \leftarrow H(nam_c, nil, E, nil, \texttt{type}, nil, \texttt{rdf:langString})$.

For each $[nil].s \in V_{CS}$, where s is not a well-typed string:
(19) $\texttt{false} \leftarrow H(nam_c, nil, s, nil, \texttt{type}, nil, \texttt{xsd:string})$.

For each RDF and RDFS axiomatic triple (s, p, o) s.t. $p, s, o \in V_{RDF}^{\#n_{CS}} \cup \mathcal{V}_{RDFS}$:
(20) $H(nam_c, nil, s, nil, p, nil, o) \leftarrow \texttt{true}$.

c-RDF graph rules
For each $([u_s].s, [u_p].p, [u_o].o) \in compl\_graph(c)$, where $c \in CS$:
(1) $H(nam_c, u_s, s, u_p, p, u_o, o) \leftarrow \texttt{true}$.

We are now ready to define the definite logic program P_{CS}:

$$P_{CS} = \bigcup \{P_c \mid c \in CS\}$$

Note that regarding the c-simple interpretation rules, if the c-RDF tuple $([u_x].x, [u_y].y, [u_z].z)$ holds in a context c' then it is derived from the c-RDFS interpretation condition 1 that (i) $([u_x].x, nil.\texttt{type}, nil.\texttt{Resource})$ holds also in c', (ii) $([u_z].z, nil.\texttt{type}, nil.\texttt{Resource})$ holds also in c' and (iii) $([u_y].y, nil.\texttt{type}, nil.\texttt{Property})$ holds also in c'. The c-RDFS interpretation rules follow from the rest of the c-RDFS interpretation conditions. In particular, note that traversing contexts rules (1), (2), (3), and (4) correspond to the c-RDFS interpretation conditions 7, 9, 11, and 13, respectively. Additionally, note that P_{CS} contains constraints, that is rules with head `false`. Basically, the c-RDFS interpretation rule (17) and the constraints (18) and (19) correspond to c-RDFS interpretation conditions 19 and 20. Additionally, the c-RDFS interpretation rule (16) corresponds to c-RDFS interpretation condition 16.

The following proposition holds determining if a contextual structure CS is c-RDFS consistent:

Proposition 2. Let CS be a contextual structure. It holds that $\mathcal{M}(CS) = \emptyset$ iff $\texttt{false} \in T_{P_{CS}}^{\uparrow\omega}(\emptyset)$.[8] □

Thus, if $\texttt{false} \in T_{P_{CS}}^{\uparrow\omega}(\emptyset)$, which is caused due to ill typing to the datatypes in D, then there is no c-RDF interpretation that satisfies CS. Note that in the RDF 1.1 semantics [14], it is not specified when an RDF graph has no RDFS interpretations.

In the rest of the paper, we consider that $\texttt{false} \notin T_{P_{CS}}^{\uparrow\omega}(\emptyset)$.

Using P_{CS}, we will construct a contextual structure, called the *closure* of CS and denoted by $cl(CS)$ such that $cl(CS) = \{cl(c, CS) \mid c \in CS\}$. Let $c \in CS$. The context $cl(c, CS)$ is defined as follows: $cl(c, CS) = \langle nam_c, G_{cl(c,CS)}, ref_c \rangle$, where $G_{cl(c,CS)} = \{([u_s].s, [u_o].p, [u_o].o) \mid H(nam_c, u_s, s, u_p, p, u_o, o) \in T_{P_{CS}}^{\uparrow\omega}(\emptyset)$ and $p \notin BN\}$.

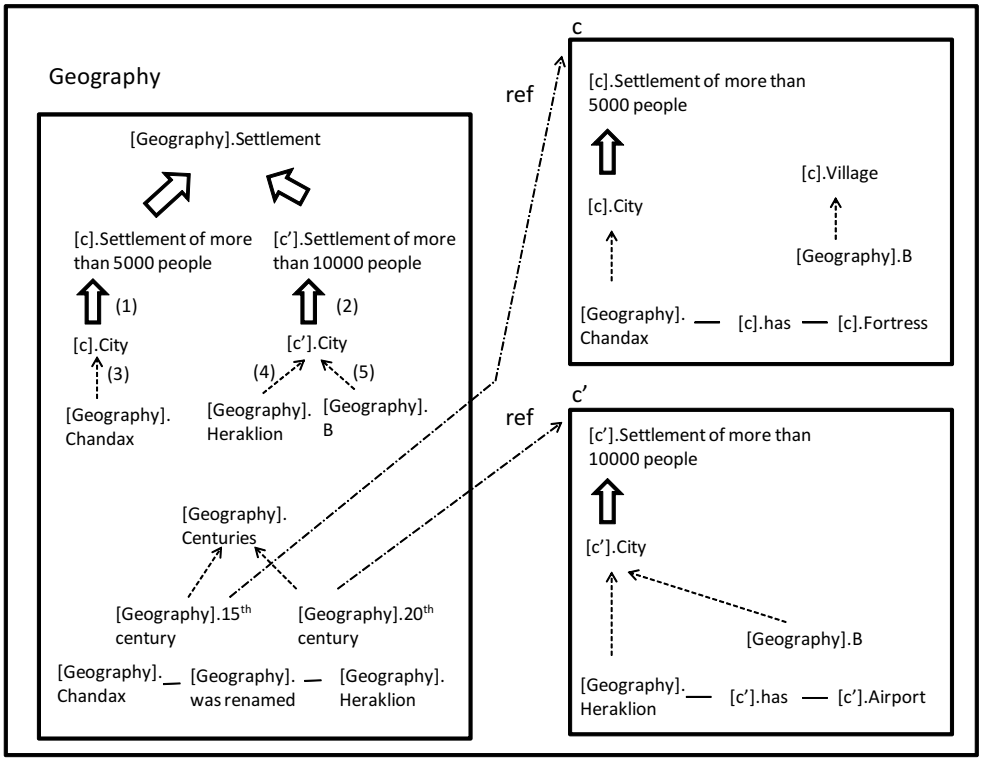

Fig. 5. Part of the closure of the contextual structure CS depicted in Figure 1.

Example 3. Note that part of the closure of the contextual structure CS depicted in Figure 1 is provided in Figure 5, where, for simplification reasons, we have considered the application of only the traversing contexts rules and and the c-RDF graph rules. Note that all c-RDF graphs in the figure contain only contextual URIs. Further note that through the use of traversing contexts rule (2), the `subClassOf` relationships (1) and (2) in the context named `Geography` are derived. Additionally, that through the use of traversing contexts rule (1), the `type` relationships (3), (4), and (5) in the context named `Geography` are derived. □

[8] $T_P(.)$ is the *immediate consequence operator* on a definite logic program P and is defined in [20]. $T_P^{\uparrow\omega}(\emptyset)$ denotes that the operator T_P is applied from the empty set until closure.

We now provide a method for checking entailment between two contextual structures.

First, we provide a few auxiliary definitions. We define $\texttt{c-}BN_c = \{x \in \texttt{c-}BN$ appearing in $compl\_graph(c)\}$. Additionally, we define $\texttt{c-}VBN_{CS} = V_{CS} \cup \{x \in \texttt{c-}BN$ appearing in $compl\_graph(c) \mid c \in CS\}$. Let c be a context and μ be a mapping from the set $\texttt{c-}BN_c$ to a set V. We denote by $\mu(c)$ the c-RDF graph that is derived if we replace each $x \in \texttt{c-}BN_c$ appearing in $compl\_graph(c)$ by $\mu(x)$.

Proposition 3 (method for entailment between two contextual structures). Let CS, CS' be two contextual structures such that $nam_{CS'} \subseteq nam_{CS}$ and $n_{CS'} \leq n_{CS}$. Let m be a mapping from CS' to CS such that if $m(c) = c'$, it holds that $nam_c = nam_{c'}$. Then, $CS \models CS'$ iff (i) there is a mapping μ from $\bigcup_{c \in CS'} \texttt{c-}BN_c$ to $\texttt{c-}VBN_{CS}$ such that $\mu(c) \subseteq G_{cl(m(c),CS)}$, for all $c \in CS'$, and (ii) if $ref_c(u) = u'$, for $c \in CS'$, then $ref_{m(c)}(u) = u'$. □

Example 4. Let CS and CS' be the contextual structures depicted in Figures 1 and 2, respectively. Let c_1 and c_2 be the contexts named `Geography` and `c` of contextual structure CS, respectively. Additionally, let c'_1 and c'_2 be the contexts named `Geography` and `c` of contextual structure CS', respectively. Note that $\texttt{c-}BN_{c'_1} = [nil].\_ : x$ and $\texttt{c-}BN_{c'_2} = \emptyset$. Let μ be a mapping from $\texttt{c-}BN_{c'_1} \cup \texttt{c-}BN_{c'_2}$ to $\texttt{c-}VBN_{CS}$ such that $\mu([nil].\_ : x) = [\texttt{Geography}].\texttt{Heraklion}$. Then, $\mu(c'_1) \subseteq G_{cl(c_1,CS)}$ and $\mu(c'_2) \subseteq G_{cl(c_2,CS)}$. Additionally, it holds that (i) if $ref_{c'_1}(u) = u'$ then $ref_{c_1}(u) = u'$ and (ii) $ref_{c'_2}(u) = u'$ then $ref_{c_2}(u) = u'$. Therefore, $CS \models CS'$. □

The following proposition shows that the problem of entailment between two contextual structures is not harder than entailment between RDF graphs.

Proposition 4. Let CS, CS' be contextual structures such that $n_{CS'} \leq n_{CS}$. Checking if $CS \models CS'$ has NP-complete time complexity. □

As the following propositions show, the contextual structures CS and $cl(CS)$ have the same entailments and one entail the other.

Proposition 5. Let CS and CS' be contextual structures such that $n_{CS'} \leq n_{CS}$. It holds that:
$CS \models CS'$ iff $cl(CS) \models CS'$. □

From this, the following corollary follows:

Corollary 1. Let CS be a contextual structure. It holds that:
(i) $cl(CS) \models CS$ and (ii) $CS \models cl(CS)$. □

The following proposition indicates that $cl(CS)$ is the largest contextual structure entailed by CS.

Proposition 6. Let CS be a contextual structure and let $c \in CS$. Let t be a c-RDF triple consisting of contextual names that are not `rdf:_i` terms such that $t \notin G_{cl(c,CS)}$. Let c' be a new context such that $c' = \langle nam_c, G_{cl(c,CS)} \cup \{t\}, ref_c \rangle$. Let $CS' = (cl(CS) \setminus cl(c, CS)) \cup \{c'\}$. Then, it holds that $CS \not\models CS'$. □

Note that the function ref_c, for $c \in CS$, equals the function $ref_{cl(c,CS)}$.

5 Merging of contexts

Let CS be a conceptual structure, it is many times desirable to incorporate a context $c_2 \in CS$ into another context $c_1 \in CS$, possibly because c_2 contains a partial view of the subject treated in c_1. The following Algorithm incorporates context c_2 into context c_1 and returns a new contextual structure such that this incorporation has taken place. Of source, since context c_2 has been incorporated into context c_1 all of the subcontexts of c_2 are also incorporated into the corresponding subcontexts of c_1.

Algorithm 1 *context_incorpor*(c_1, c_2, CS)
Input: A contextual stucture CS such that $c_2 \in CS$ is to be incorporated into context $c_1 \in CS$
Output: A new contextual structure after the incorporation has taken place

(1) Let $new\_graph = compl\_graph(c_1) \cup compl\_graph(c_2)$;
(2) Replace in the new graph all $[nam_{c_1}].u$ and $[nam_{c_2}].u$ by u.
(3) Let $CS_{new} = CS$;
(4) For each $u \in IRI$ such that $ref_{c_1}(u)$ or $ref_{c_2}(u)$ is defined then
(5) if $ref_{c_1}(u)$ is defined and $ref_{c_2}(u)$ is not defined then
(6) Let $ref(u) = ref_{c_1}(u)$;
(7) if $ref_{c_2}(u)$ is defined and $ref_{c_1}(u)$ is not defined then
(8) Let $ref(u) = ref_{c_2}(u)$;
(9) if both $ref_{c_2}(u)$ and $ref_{c_1}(u)$ are defined then
(10) Let $ref(u) = ref_{c_1}(u)$;
(11) $CS_{old} = CS_{new}$;
(12) Let $CS_{new} = context\_incorpor(nam^{-1}(ref_{c_1}(u)), nam^{-1}(ref_{c_2}(u)), CS_{old})$.
(13) *End for*
(14) If there is no $u \in IRI$ such that $ref_{c_1}(u)$ or $ref_{c_2}(u)$ are defined then ref is the empty mapping;
(15) Let $c = \langle nam_{c_1}, new\_graph, ref \rangle$;
(16) Return$((CS_{new} \setminus \{c_1\}) \cup \{c\}$.

In Algorithm *context_incorpor*(c_1, c_2, CS) first the c-RDF graph of context c_1 is combined with the c-RDF graph of context c_2, creating a new c-RDF graph *new_graph*. Since context c_2 is incorporated in context c_1, all contextual IRIs $[nam_{c_1}].u$ and $[nam_{c_2}].u$ in *new_graph* are replaced by u. In other words, all IRIs local in c_2 are becoming local to the new context created after the incorporation. Then, the new function $ref : IRIs \rightarrow CNAM - \{nil\}$ is defined. In particular, for each IRI u such that $ref_{c_1}(u)$ or $ref_{c_2}(u)$ is defined the new function ref is defined as follows. If $ref_{c_1}(u)$ is defined then $ref(u) = ref_{c_1}(u)$. If, however, $ref_{c_1}(u)$ is not defined then $ref(u) = ref_{c_2}(u)$. In the case that both $ref_{c_1}(u)$ and $ref_{c_2}(u)$ are defined then the context named $ref_{c_2}(u)$ is recursively incorporated into the context named $ref_{c_1}(u)$. This process is continued recursively until all subcontexts of c_2 have been incorporated into the corresponding subcontexts of c_1. Then, the new contextual structure which replaces the context c_1 by the context $c = \langle nam_{c_1}, new\_graph, ref \rangle$ is returned. Note that both c and c_1 have the same name.

In particular, if a context c is reached from context c_1 through a sequence of IRIs $u_1.u_2....u_n$ (indicating its semantics) and a context c' is reached from context c_2 through the same sequence of IRIs then the contents of c and c' are combined creating a new context c'' with the same name as c which replaces c in the contextual structure. The new context c'' is reached though the same sequence of IRIs from the new context that replaces c_1 in the contextual structual stucture (after merging with c_2).

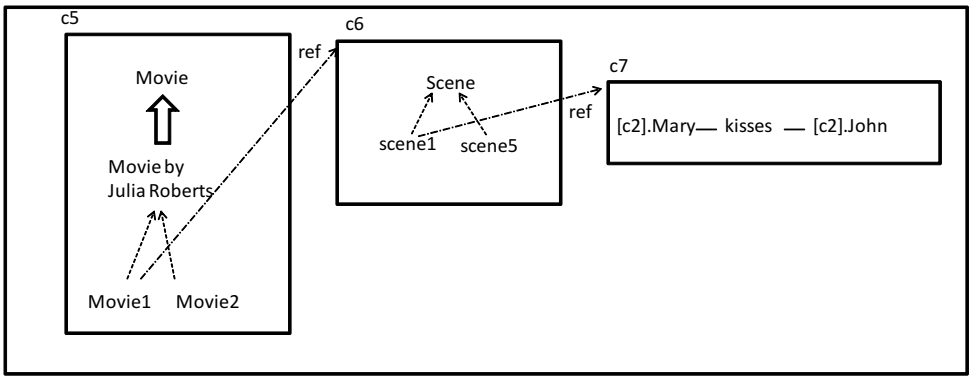

Fig. 6. Enlargement of the contextual structure of Figure 4

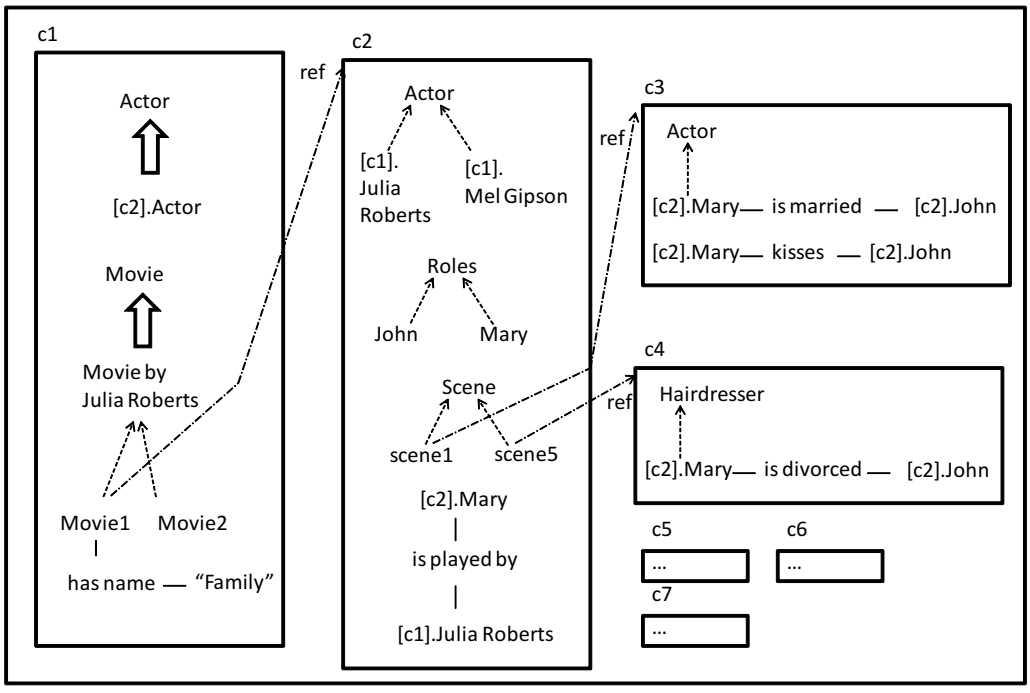

Fig. 7. Merging of contexts

Example 5. Assume that in the contextual structure of Figure 4, we add the nested contexts of Figure 6, creating an enlarged contextual structure CS. Further, assume

that the contexts named ci correspond to contexts c_i and that we now call the Algorithm *context_incorpor*(c_1, c_5, CS). Then, the result is the contextual structure CS' shown in Figure 7. Note that the contents of context c_5 have been incorporated in the contents of context c_1, showing that `Movie by Julia Roberts` is a subclass of `Movie` and it has the instances `Movie1` and `Movie2`. Recursively, the contents of context c_6 have been incorporated into the contents of context of c_2 and the contents of context c_7 have been incorporated into the contents of context of c_3. Note that the contexts after the incorporation keep their original names. Thus, in the new context named `c3` of contextual structure CS', the person with role Mary in the movie called *Family* and in scene 1 kisses the person with role John. □

The following proposition provides the complexity of Algorithm *context_incorpor*(c_1, c_2, CS).

Proposition 7. Let CS be a contextual structure and $c_1, c_2 \in CS$. The complexity of the algorithm *context_incorpor*(c_1, c_2, CS) is in $O(n_{c_1} * S_{max})$, where n_{c_1} is the number of subcontexts of c_1 and S_{max} is the size of the largest context c_1, c_2 and their subcontexts (the size is in terms of the number of c-RDF triples and the number of references of the contexts).

6 Query language

The c-SPARQL language for querying contexts extends the SPARQL 1.0 language [25] for querying RDF graphs. Similarly to SPARQL, c-SPARQL queries are defined by graph patterns, which are obtained by combining triple patterns with operators. A simple variable in c-SPARQL is a term prefixed by "?"'. The set of simple variables is denoted by *Var*. A *complex contextual variable* has the form $[y].x$, where (i) y is in *CNAM* or y is a simple variable and (ii) x is a name or blank node or a simple variable. However, at least one of y and x must be a simple variable. If y is a simple variable then it is called *context variable*. If x is a simple variable then it is called *body variable*. The set of complex contextual variables is denoted by c-*Var*.

(c-*SPARQL*) *graph patterns* are defined recursively as follows:

- A triple (s, p, o), where $s, o \in Nam \cup$ c-$Nam \cup Var \cup$ c-Var and $p \in IRI \cup$ c-$IRI \cup Var \cup$ c-Var is a graph pattern, called *triple pattern*;
- If P_1 and P_2 are graph patterns then (P_1 `AND` P_2), (P_1 `UNION` P_2), and (P_1 `OPTIONAL` P_2) are graph patterns;
- If P_1 is a graph pattern and R is a filter SPARQL expression[9] then the construction (P_1 `FILTER` R) is a graph pattern;
- If P_1 and P_2 are graph patterns and R is a filter SPARQL expression then (P_1 `OPTIONAL` (P_2 `FILTER` R)) is a graph pattern;
- If P_1 is a graph pattern and $cn \in Var \cup CNAM$ then (`CONTEXT` cn P_1) is a graph pattern.
- If P_1 is a graph pattern and u is a variable or an IRI then (`CONTEXT REF` u P_1) is a graph pattern.

[9] For the full syntax of filter expressions, see the W3C recommendation [25].

The set of simple variables of a c-SPARQL graph pattern P is denoted by $Var(P)$.

Evaluation of c-SPARQL graph patterns returns multisets (bags) of solution mappings. A *solution mapping*, abbreviated *solution*, is a partial function $\mu : Var \rightarrow Nam \cup \text{c-}Nam \cup BN \cup \text{c-}BN$. The domain of μ is the subset of variables of Var, where μ is defined. Two mappings μ_1 and μ_2 are compatible if for every variable v in $dom(\mu_1) \cap dom(\mu_2)$ it is the case that $\mu_1(v) = \mu_2(v)$. It is important to understand that any mappings with disjoint domains are compatible. If two solutions μ_1 and μ_2 are compatible then their union $\mu_1 \cup \mu_2$ is also a solution mapping. We represent extensionally a solution mapping μ as a set of pairs of the form (v, t), where $u \in Var$ and $t = \mu(u)$; in the case of a solution mapping with a singleton domain we use the abbreviation $v \rightarrow t$.

We denote that a solution mapping μ satisfies the filter expression R by $\mu \models R$.

Definition 13 (evaluation SPARQL algebra operators [25]). Let Ω_1 and Ω_2 be multisets of solution mappings, and R a filter expression. Define:

Join: $\Omega_1 \bowtie \Omega_2 = \{|\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1 \text{ and } \mu_2 \in \Omega_2 \text{ such that } \mu_1 \text{ and } \mu_2 \text{ are compatible } |\}$
Union: $\Omega_1 \cup \Omega_2 = \{|\mu \mid \mu \in \Omega_1 \text{ or } \mu \in \Omega_2|\}$
Diff: $\Omega_1 \setminus_R \Omega_2 = \{|\mu_1 \mid \mu_1 \in \Omega_1$ such that $\forall_{\mu_2 \in \Omega_2}$ either μ_1 and μ_2 are not compatible, or μ_1 and μ_2 are compatible and $\mu_1 \cup \mu_2 \not\models R|\}$
LeftJoin: $\Omega_1 \; \text{⟕}_R^{D(G)} \; \Omega_2 = (\Omega_1 \bowtie \Omega_2) \cup (\Omega_1 \setminus_R \Omega_2)$ □

Note that the **Diff** operator is auxiliary to the definition of **LeftJoin**.

Let t a c-RDF triple and let c be a context, we denote by $complete_c(t)$ the c-RDF triple that is derived if (i) we replace each non-contextual IRI w appearing in t that is not in $\mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup D$ by $[nam_c].u$ and (ii) we replace each non-contextual name and blank node w in $\mathcal{V}_{RDF} \cup \mathcal{V}_{RDFS} \cup \mathcal{LIT} \cup D \cup BN$ appearing in t by $[nil].w$.

Below be define the evaluation of the c-SPARQL algebra operators.

Definition 14 (c-SPARQL graph pattern evaluation). Let $CS(c)$ be a conceptual structure with active context $c \in CS$. Let P, P_1, and P_2 be arbitrary graph patterns, and t be a triple pattern. The evaluation of a graph pattern over $CS(c)$, denoted by $[\![.]\!]_{CS(c)}$ is defined recursively as follows:

1. $[\![t]\!]_{CS(c)} = \{|\ \mu \mid dom(\mu) = var(t)$ and $complete_c(\mu(t)) \in G_{cl(c,CS)}|\}$, where $var(t)$ is the set of simple variables occurring in the triple pattern t;
2. $[\![(P_1 \texttt{ AND } P_2)]\!]_{CS(c)} = [\![P_1]\!]_{CS(c)} \bowtie [\![P_2]\!]_{CS(c)}$;
3. $[\![(P_1 \texttt{ UNION } P_2)]\!]_{CS(c)} = [\![P_1]\!]_{CS(c)} \cup [\![P_2]\!]_{CS(c)}$;
4. $[\![(P_1 \texttt{ OPTIONAL } P_2)]\!]_{CS(c)} = [\![P_1]\!]_{CS(c)} \; \text{⟕}_{true} \; [\![P_2]\!]_{CS(c)}$;
5. $[\![(P_1 \texttt{ OPTIONAL } (P_2 \texttt{ FILTER } R))]\!]_{CS(c)} = [\![P_1]\!]_{CS(c)} \; \text{⟕}_R \; [\![P_2]\!]_{CS(c)}$;
6. $[\![(P_1 \texttt{ FILTER } R)]\!]_{CS(c)} = \{|\ \mu \in [\![P_1]\!]_{CS(c)} \mid \mu \models R\ |\}$;
7. Evaluation of $[\![(\texttt{CONTEXT } term\ P_1)]\!]_{CS(c)}$ depends on the form of $term$:
   - If $term$ is an IRI corresponding to the name of a context $c' \in CS$ then $[\![(\texttt{CONTEXT } term\ P_1)]\!]_{CS(c)} = [\![P_1]\!]_{CS(c')}$;
   - If $term$ is an IRI that does not correspond to any context in CS then $[\![(\texttt{CONTEXT } term\ P_1)]\!]_{CS(c)} = \{|\}$;
   - If $term$ is a variable v and $CS = \{c_1, ..., c_n\}$ then $[\![(\texttt{CONTEXT } term\ P_1)]\!]_{CS(c)} =$

$$= ([\![P_1]\!]_{CS(c_1)} \bowtie \{|v \rightarrow nam_{c_1}|\}) \cup \ldots \cup ([\![P_1]\!]_{CS(c_n)} \bowtie \{|v \rightarrow nam_{c_n}|\})$$

8. Evaluation of $[\![(\texttt{CONTEXT REF}\ term\ P_1)]\!]_{CS(c)}$ depends on the form of $term$:
   - If $term$ is an IRI such that $ref_c(term)$ is defined, corresponding to context c', then $[\![(\texttt{CONTEXT REF}\ term\ P_1)]\!]_{CS(c)} = [\![P_1]\!]_{CS(c')}$;
   - If $term$ is an IRI such that $ref_c(term)$ is not defined then $[\![(\texttt{CONTEXT REF}\ term\ P_1)]\!]_{CS(c)} = \{\!|\}$;
   - If $term$ is a variable v, ref_c is defined in the IRIs $\{u_1, ..., u_n\}$, and $ref_c(u_i)$ corresponds to a context c_i then $[\![(\texttt{CONTEXT REF}\ term\ P_1)]\!]_{CS(c)} =$

$$= ([\![P_1]\!]_{CS(c_1)} \bowtie \{\!|v \to u_1|\!\}) \cup \ldots \cup ([\![P_1]\!]_{CS(c_n)} \bowtie \{\!|v \to u_n|\!\})$$

□

The evaluation of the c-SPARQL graph pattern $[\![\texttt{CONTEXT}\ term\ P]\!]_{CS(c)}$ depends on the cases (i) $term$ is an IRI corresponding to the name of a context $c' \in CS$ and (ii) $term$ is a variable. In case (i), the graph pattern P is evaluated in context c'. In case (ii), the graph pattern P is evaluated in all contexts of CS. On the other hand, the evaluation of the c-SPARQL graph pattern $[\![\texttt{CONTEXT REF}\ term\ P]\!]_{CS(c)}$ depends on the cases (i) $term$ is an IRI and (ii) $term$ is a variable. In case (i), the graph pattern P is evaluated in the context $ref_c(term)$. In case (ii), the graph pattern P is evaluated in all contexts $ref_c(u)$, where $ref_c(u)$ is defined.

Below, we present Algorithm 2 that takes as input a mapping μ on the simple variables of the c-SPARQL graph pattern P and a contextual structure CS with active context $c \in CS$ and returns TRUE if $\mu \in [\![P]\!]_{CS(c)}$, and FALSE otherwise.

As an auxillary definition we now define the set of mappings $POS(P, CS)$, where P is a graph pattern and CS a conceptual structure. In particular, $POS(P, CS)$ is the set of partial functions v from $Var(P)$ to context names, names, blank nodes, contextual names, and contextual blank nodes appearing in CS with the following constraints: (i) a contextual variable $?cn$ is mapped to a context name in CS and (ii) body variable $?w$ is mapped to a name or blank node in CS.

Algorithm 2 *Eval*$(\mu, P, CS(c))$
Input: A mapping μ on the simple variables of the c-SPARQL graph pattern P and a contextual structure CS with active context $c \in CS$
Output: TRUE if $\mu \in [\![P]\!]_{CS(c)}$, and FALSE otherwise

Case:
(1) P is a triple pattern t
(2) If $var(\mu) = var(t)$ and $complete_c(\mu(t)) \in G_{cl(c,CS)}$ then *return*(TRUE);
(3) P is a pattern of the form (P_1 FILTER R);
(4) if $Eval(\mu, P_1, CS(c))$ = TRUE and $\mu \models R$ then *return*(TRUE);
(5) P is a pattern of the form (P_1 UNION P_2);
(6) if $Eval(\mu, P_1, CS(c))$ = TRUE or $Eval(\mu, P_2, CS(c))$ = TRUE then
(7) *return*(TRUE);
(8) P is a pattern of the form (P_1 AND P_2);
(10) If it exists $\mu_1 \in POS(P_1, CS)$ such that $Eval(\mu_1, P_1, CS(c))$ and
(11) it exists $\mu_2 \in POS(P_2, CS)$ such that $Eval(\mu_2, P_2, CS(c))$ and $\mu = \mu_1 \cup \mu_2$
(12) then *return*(TRUE);
(13) P is a pattern of the form (P_1 OPTIONAL P_2);
(14) If $Eval(\mu, P_1$ AND $P_2)$=TRUE then *return*(TRUE);

(15) If *Eval*($\mu, P_1,\ CS(c)$)=TRUE then
(16) if for each mapping $\mu' \in POS(P_2, CS)$ such that *Eval*($\mu', P_2,\ CS(c)$)=TRUE,
(17) it holds that μ and μ' are not compatible then *return*(TRUE);
(18) *return*(FALSE);
(19) P is a pattern of the form (P_1 `OPTIONAL` (P_2 `FILTER` R));
(20) If *Eval*(μ, P_1 `AND` P_2)=TRUE then *return*(TRUE);
(21) If *Eval*($\mu, P_1,\ CS(c)$)=TRUE then
(22) if for each mapping $\mu' \in POS(P_2, CS)$ such that *Eval*($\mu', P_2,\ CS(c)$)=TRUE,
(23) it holds that μ and μ' are not compatible or $\mu \cup \mu' \not\models R$ then *return*(TRUE);
(24) *return*(FALSE);
(25) P is a pattern of the form (`CONTEXT` cn P), where cn is a variable ;
(26) if it exists a $\mu_1 \in POS(P, CS)$ such that *Eval*($\mu_1, P,\ CS(\mu(cn))$)=TRUE and
(27) $\mu = \mu_1 \cup (cn \rightarrow \mu(cn))$;
(28) then *return*(TRUE);
(29) P is a pattern of the form (`CONTEXT` cn P), where $cn \in CNAM$;
(30) *return*(*Eval*($\mu, P,\ CS(cn)$));
(31) P is a pattern of the form (`CONTEXT REF` u P), where u is a variable ;
(32) if it exists a $\mu_1 \in POS(P, CS)$ such that *Eval*($\mu_1, P,\ CS(ref_c(\mu(u)))$)=TRUE and
(33) $\mu = \mu_1 \cup (u \rightarrow \mu(u))$;
(35) then *return*(TRUE);
(35) P is a pattern of the form (`CONTEXT REF` u P), where $u \in IRI$;
(36) If $ref_c(u)$ is defined then *return*(*Eval*($\mu, P,\ CS(ref_c(u))$));
(37) else *return*(FALSE);
(38) *return*(FALSE);

The explanation of Algorithm 2 is straightforward as it follows exactly Definition 14. In particular, line (1) follows from Definition 14 item 1. Line (3) follows from Definition 14 item 6. Line (5) follows from Definition 14 item 3. Line (8) follows from Definition 14 item 2. Line (13) follows from Definition 14 item 4. Line (19) follows from Definition 14 item 5. Lines (25) and (29) follow from Definition 14 item 7. Lines (31) and (35) follow from Definition 14 item 8. Algorithm 2 extends the Algorithm *Eval*(μ:mapping, P:SPARQL graph pattern, G: RDF graph) as provided in [24].

Obviously, Algorithm 2 is in PSPACE in the size of CS and P. Since c-SPARQL extends SPARQL and the complexity of SPARQL operating on a graph G has combined PSPACE-complete [24], Algorithm 2 is PSPACE-hard in the size of CS and P. Thus, the complexity of c-SPARQL operating on a contextual structure CS has combined complexity PSPACE-complete.

Below, we define the syntax of c-SPARQL as a modification of the syntax of SPARQL [25].

In particular, we replace:

```
SelectQuery ::=  SELECT (  DISTINCT |  REDUCED )? ( Var+ | '*' )
                 DatasetClause* WhereClause SolutionModifier
```

by

```
SelectQuery ::=  SELECT  (  DISTINCT |  REDUCED )?
                 ( Var+ |CVar+ | REF CVar | '*' )
                FromContextClause WhereClause SolutionModifier
CVar=  [ Var ]. Var  |   [ IRIref ]. Var  |  [ Var ]. IRIref
FromContextClause :=  FROM   IRIref  CONTEXTUAL STRUCTURE  IRIref
```

We replace:

```
GraphGraphPattern ::=  GRAPH   VarOrIRIref GroupGraphPattern
```
by
```
GraphGraphPattern ::=  CONTEXT   VarOrIRIref GroupGraphPattern  |
                  CONTEXT REF  VarOrIRIref GroupGraphPattern
```

We replace:
```
VarOrTerm ::= Var | GraphTerm
```

by
```
VarOrTerm ::= CVar |  [ URIref  ]. URIref  |  GraphTerm
```

That is the only difference is that in the `FromContextClause`, we indicate the context c within a contextual structure CS from which navigation starts. Additionally, with the clause `GraphGraphPattern`, we achieve movement to any context within CS or to a reference of an IRI w.r.t. the current context. Variables now can have the form indicated by the clause `CVar`. Further, to the terms, we have added `[URIref].URIref` corresponding to contextual IRIs.

Example 6. Consider the contextual structure CS' of Figure 7. We here provide example queries of this contextual structure, which are actually queries on $cl(CS')$.

Query 1: Which persons work as actors?

```
Select ?x
FROM c1 CONCEPTUAL STUCTURE CS'
WHERE {?x type Actor.}
```

Answer={(?x=`[c1].Julia Roberts`, ?x=`[c1].Mel Gipson`}.

Query 2: Which are the roles played by Julia Roberts?

```
Select ?z
FROM c1 CONCEPTUAL STUCTURE CS'
WHERE {?y type Movie by Julia Roberts.
       CONTEXT REF ?y {?z is played by [?w].Julia Roberts.}}
```

Answer={(?z=`[c].Mary`)}.

Query 3: In which scene of `Movie1`, there is a role that represents an actor?

```
Select ?y
FROM c1 CONCEPTUAL STUCTURE CS'
WHERE {CONTEXT REF Movie1 {?y type Scene. ?z type Role.
     CONTEXT REF ?y {?z type Actor.}}}
```

Answer={(?y=`[c2].scene1`)}.

Query 4: In which movie and which scenes Julia Roberts is married to a man and then is getting divorced?

```
Select ?x, ?y1,?y2
FROM c1 CONCEPTUAL STUCTURE CS'
WHERE {?x type Movie. CONTEXT REF ?x {?y1 type Scene.
```

```
?y2 type Scene. ?z is played by [c1].Julia Roberts.
CONTEXT REF ?y1 {?z is married ?w.}
CONTEXT REF ?y2 {?z is divorced ?w.}}
```

Answer={(?x=[c1].Movie1, , ?y1=[c2].scene1, ?y2=[c2].scene5)}.

Note that from a given context c, we can reach any resource that belongs to the reference of a resource within c, and recursively any context that lies on the path of subcontexts.

7 Related Work

Below, we review related work.

Our work has been inspired from [3], which presents a general framework for representing the notion of context in information modeling. First, the authors define a context as a set of objects, within which each object has a set of names and possibly a reference: the reference of the object is another context which "hides" detailed information about the object. Then, the possibility of structuring the contents of a context through the traditional abstraction mechanisms, i.e. classification, generalization, and attribution, is introduced. A main difference between this work and that of [3] is that [3] is based on the TELOS conceptual model [23], where every triple expressing an association (s, p, o) is also an object o' with $from(o') = s$ and $to(o') = o$. As every association (including classification and generalizations relationships) is also an object it can also have references. This is not possible in our model, since we extend RDF. In this sense, [3] is more powerful. Yet, [3] does not support the c-RDFS interpretation conditions 7, 9, 11, and 13, since it does not derive new information through spanning of contexts. Additionally, it does not support merging and entailment between contextual structures, neither a query language is proposed.

A query language is defined for [3] in [26], which focuses on the following issues: (i) accessing information in the contextual structure using paths of names, or paths of references and (ii) retrieval of contextual information. They achieve this by defining useful fundamental query operations on contexts such as select, project, generate (which allow the reorganization of context structures), and path select. The complexity of the proposed query language has not investigated. In our case, we have a language with formal syntax that extends SPARQL. Note that accessing information through contexts or paths of references is also allowed in our query language. The complexity of the proposed language is the same as the combined complexity of SPARQL, that is PSPACE-complete.

A framework for contextual RDF knowledge is considered in [16], where contextual RDF bases are organized in a partial order based on the values of metadata parameters. Similarly to our case, [16] allows associations between contextual resources and classification of a contextual resource to a contextual class. However, it does not support the c-RDFS interpretation conditions 7 and 11. Instead, [16] has the rules (i) the extension of a contextual class or property to a context c remains the same in all broader contexts and (ii) the extension of a contextual class or property in a narrower context is obtained by restricting the extension of the same class or property in the broader context to the resources of a narrower context. Of source these rules make sense only in the case that contextual RDF bases are partially ordered and not in our general case. Our c-RDFS interpretation conditions 9 and 13

that concern selected transitivity of `subClassOf` and `subPropertyOf` relations only between contexts that are related through these relations, are not considered at all. Additionally, it does not support merging and nesting of context, neither a query language is proposed.

In [11], similarly to [22], the special symbol *ist* is used and $ist(c, \phi)$ states that the proposition ϕ is true in the context c. The work in [11] uses metalevel statements through *ist* to express selective importing between contexts, preference rules, mapping constants, and mapping more complex graph patterns. These high level rules can also be applied to our framework and then compute the new closure to get all inferences. Further, this work proposes a model theory for context in the Semantic Web. Yet, this consists of a single interpretation which extend the simple interpretation of the RDFS model theory [14] and not of multiple ones as in our work and [16]. Therefore, our `c`-RDFS interpretation conditions are not considered at all. Further, [11] does not support blank nodes and it does not propose a query language.

Jie Bao et.al. [4] provide a more concrete formalization of the theory of context by McCarthy. They extend/modify the theory described in [22] with a predicate $isin(c, \phi)$ representing the fact that the statement ϕ can be interpreted in the context c. This approach is particularly suited for manipulating contexts as first class objects. A considerable number of constructs were introduced for combining contexts (like $c_1 \wedge c_2$, $c_1 \vee c_2$ and $\neg c$) and for relating contexts (like $c_1 \Rightarrow c_2$, and $c_1 \rightarrow \neg c_2$). The work, however, does not give an axiomatization of all these operators, neither a query language is proposed.

In [1], an ERDF ontology is defined as the combination of (i) an ERDF graph G containing (implicitly existentially quantified) positive and negative information, and (ii) an ERDF program P containing derivation rules, with possibly all connectives $\sim$, $\neg$, $\supset$, $\wedge$, $\vee$, $\forall$, $\exists$ in the body of a rule, and strong negation $\neg$ in the head of a rule. In [2], we propose a framework for modular ERDF ontologies, called *modular ERDF framework*, in which a modular ERDF ontology $\mathcal{R}$ is a set of `r`-ERDF ontologies. Intuitively, an `r`-ERDF ontology $O \in \mathcal{R}$ is an ERDF ontology that can import or just reference knowledge about a property or class x from other `r`-ERDF ontologies in $\mathcal{R}$ that define x. In the present work, we can simulate this importing by the RDFS interpretations rules 7 and 11. But the RDFS interpretations rules 9 and 13 have not parallel in the modular ERDF framework. Additionally, the modular ERDF framework does not support contextualized named and references pointing to nested contextualized views. The query language of the modular ERDF framework is powerful enough to support queries to `r`-ERDF ontologies using all connectives $\sim$, $\neg$, $\supset$, $\wedge$, $\vee$, $\forall$, $\exists$. However, it does not support an extension of the SPARQL language that allows navigation through contexts that can be variables and navigation through references.

Carroll et al. [7, 6] identify the problem of provenance of RDF triples. They propose *named graphs*, as an entity denoting a collection of RDF triples, which can be annotated with relevant provenance information. In particular, they do not specify how provenance itself should be represented, but they simply offer a placeholder for its representation. In our case, a context can be easily associated with metadata information.

MacGregor and Ko [21], replace each RDF triple by a quadruple, where the fourth element is the name of the context that the RDF triple is true. Additionally, provide a simple query example where query triples are replaced by query quads and variables

can be in context places. Yet, no axiomatization is presented. Additionally, the query language has no formal definition, neither it extends SPARQL.

Notation 3 (N3) [5] provides a more human readable syntax for RDF and also extends RDF by adding numerous pre-defined constructs ("built-ins") for being able to express rules conveniently. Quoting is an important feature provided by N3. An example of a quoted N3 formula is: `Mary believes that {Joe believes that {Peter is a graduate student}}`. Obviously, quoting can be very easily expressed through our formalization, since we support nesting of contexts.

Klyne [18] implements contexts using reification and collections and illustrates how the description of a complex object or system can be built-up component-wise from contexts describing the different components. For example, a car may consist of an engine and a body. The engine may consist of mechanical fuel and electrical subsystems, while the body may consist of doors, shell, etc. In our case, complex/nested subsystems we can be very easily described through nested contexts.

In [9], an ontology context is described as a named graph composed of a set of RDF triples. An ontology context can have import relationships with other ontology contexts making possible the derivation of new RDF triples through the RDFS inference triples. The *import closure* of an ontology context c consists of all the context that directly of indirectly imported in c. In our case, we define merging of contexts and their subcontexts within a contextual structure. Then, on the result another merging can be applied.

In [17], reified RDF statements A are associated through the property *cdfs*:*trueInContext* with a collection B of other reified RDF statements stating the conditions under which A is true. Reified RDF statements C in B may also be associated through the property *cdfs*:*trueInContext* with a collection D of other reified RDF statements stating the conditions under which C is true. This way a nesting of conditions can be built. In our case, each context c may be associated with another context c' containing the conditions under which the c-RDF triples in c are true. This can be done by inserting in c a resource `metadata` that refers to c'. Additionally, we can simulate the work in [17], if each context contains reified RDF statements and instead of the *cdfs*:*trueInContext* property, we use references that point to contexts that contain other reified RDF statements, and so on, simulating the nesting of conditions.

In [10], multidimensional RDF is defined, where each RDF triple is split into three parts with the middle part indicating the conditions/metadata under which the RDF triple is true. In our case, the metadata of a context are described for the whole context. Splitting multidimensional RDF documents into contexts with the same metadata we achieve the same result. Further, the work in [10] is not presented as an extension of the RDFS theory and a query language is not presented.

Table 1 overviews comparison with related work.

8 Conclusions

In this paper, we have defined contexts extending RDF graphs and nesting of contexts leading to contextual structures. We have defined c-RDFS interpretations of contextual structures extending RDFS interpretations. Additionally, we have defined c-RDFS entailment between contextual stuctures exteding RDFS entailment between RDFS graphs. For each contextual structure CS, a contextual structure $cl(CS)$, called *closure* of CS, is defined which contains all entailments. Through the notion

features	[3]	[16]	[11]	[4]	[2]	[7]	[21]	[5]	[18]	[9]	[17]	[10]	our work
context	√	√	√	√	√	√	√	×	√	√	√	×	√
foreign contextual URIS	×	√	×	×	×	×	×	×	×	×	×	×	√
linking of contexts	√	∘	×	×	×	×	×	×	×	×	×	×	√
nested RDF triples	×	×	×	×	×	×	×	√	√	×	×	×	√
metadata	√	√	√	√	×	√	×	×	×	×	√	√	√
partial importing of knowledge between contexts	×	√	×	×	√	×	×	×	×	×	×	×	√
selective transitivity of `subClassOf` and `subPropertyOf` relations between contexts	×	×	×	×	×	×	×	×	×	×	×	×	√
extension of the RDFS 1.1 model theory	×	√	×	×	√	×	×	×	×	×	×	×	√
extension of the SPARQL 1.0 query language	×	×	×	×	×	×	×	×	×	×	×	×	√
entailment between contextual structures	×	×	×	×	×	×	×	×	×	×	×	×	√
merging of contexts and subcontexts	×	×	×	×	×	×	×	×	×	×	×	×	√

Table 1. Comparison with related work (∘ means partial)

of closure of a contextual structure, a method for checking (c-RDFS) entailment between contextual structures is defined, which has NP-complete time complexity. Thus, checking entailment between contextual structures is not harder than checking RDFS-entailment between RDF graphs.

Merging between two contexts c_1 and c_2 of a contextual structure CS is also defined which incorporates not only the context c_2 to c_1 but also merges all corresponding subcontexts of c_1 and c_2. We formally define a query language, called c-SPARQL, operating on the closure of contextual structure which extends and has the same complexity as SPARQL 1.0 operating on RDF graphs. In particular, all its algebra, evaluation, and syntax are provided.

Future work concerns the redefinition of the c-SPARQL query language such that it extends SPARQL 1.1. [13].

Evaluating our work, we satisfy all goals mentioned in the motivation of our work in the introductory section. Though some of our features are present in previous works, as summarized in Table 1 and described in detail in the related work section, our approach has some novel features such as extension of the RDFS1.1 model theory to support contextual structures. Literal values are treated in a common way even though they may be contextualized within contexts. We have the novel concept of entailment between contextual structures. Additionally, we present an extension of the SPARQL 1.0 query language on contextual structures that is able to query the contents of contexts and navigate within different contexts through references. This extension is unique to our work.

Extra expressivity is provided, namely that is possible to express general statements that in a given context, a subject under some context is related via a predicate (interpreted in a context) to a object in another context; all these contexts can be the same or different. This is particularly striking from the construction of a closure of a contextual structured used to perform inference with contextual structures,

where a septet predicated is used to represent information instead of the simpler quad approach present in current approaches.

References

1. A. Analyti, G. Antoniou, C. V. Damásio, and G. Wagner. Extended RDF as a Semantic Foundation of Rule Markup Languages. *Journal of Artificial Intelligence Research (JAIR)*, 32:37–94, 2008.
2. Anastasia Analyti, Grigoris Antoniou, Carlos Viegas Damásio, and Ioannis Pachoulakis. A Framework for Modular ERDF Ontologies. *Annals of Mathematics and Artificial Intelligence*, 67(3-4):189–249, 2013.
3. Anastasia Analyti, Manos Theodorakis, Nicolas Spyratos, and Panos Constantopoulos. Contextualization as an Independent Abstraction Mechanism for Conceptual Modeling. *Information Systems*, 32(1):24–60, 2007.
4. J. Bao, J. Tao, and D. McGuinness. Context Representation for the Semantic Web. In *Web Science Conference (WebSci10)*, 2010.
5. Tim Berners-Lee, Dan Connolly, Lalana Kagal, Yosi Scharf, and Jim Hendler. N3Logic: A Logical Framework For the World Wide Web. *Theory and Practice of Logic Programming (TPLP)*, 8(3):249–269, 2008.
6. Jeremy J. Carroll, Christian Bizer, Patrick J. Hayes, and Patrick Stickler. Named graphs. *Journal of Web Semantics*, 3(4):247–267, 2005.
7. Jeremy J. Carroll, Christian Bizer, Patrick J. Hayes, and Patrick Stickler. Named graphs, Provenance and Trust. In *14th International Conference on World Wide Web (WWW-2005)*, pages 613–622, 2005.
8. J. de Bruijn, E. Franconi, and S. Tessaris. Logical Reconstruction of Normative RDF. In *OWL: Experiences and Directions Workshop (OWLED-2005)*, Galway, Ireland, November 2005.
9. R. Delbru, A. Polleres, G. Tummarello, and S. Decker. Context Dependent Reasoning for Semantic Documents in Sindice. In *4th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS-2008)*, 2008.
10. Manolis Gergatsoulis and Pantelis D. Lilis. Multidimensional RDF. In *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, OTM Confederated International Conferences*, pages 1188–1205, 2005.
11. Ramanathan V. Guha, Rob McCool, and Richard Fikes. Contexts for the Semantic Web. In *Third International Semantic Web Conference (ISWC-2004)*, pages 32–46, 2004.
12. Claudio Gutierrez, Carlos A. Hurtado, Alberto O. Mendelzon, and Jorge Pérez. Foundations of semantic web databases. *Journal of Computer and System Sciences*, 77(3):520–541, 2011.
13. Steve Harris and Andy Seaborne. SPARQL 1.1 Query Language. W3C Recommendation, 21 March 2013. Available at `http://www.w3.org/TR/sparql11-query/`.
14. Patrick Hayes and Peter F. Patel-Schneider. RDF 1.1 Semantics. W3C Recommendation, 25 February 2014. Available at `http://www.w3.org/TR/2014/REC-rdf11-mt-20140225/`.
15. Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space (1st edition)*. Morgan & Claypool, 2011.
16. Mathew Joseph and Luciano Serafini. Simple Reasoning for Contextualized RDF Knowledge. In *Fifth International Workshop on Modular Ontologies (WoMO 2011)*, pages 79–93, 2011.
17. O. Khriyenko and V. Terziyan. Context Description Framework for the Semantic Web. In *Context Representation and Reasoning Workshop*, 2005.
18. Graham Klyne. Information Modelling using RDF - Constructs for Modular Description of Complex Systems, 2001. Available at `http://www.ninebynine.org/RDFNotes/RDFInfoModelling.pdf`.

19. Timothy Lebo, Satya Sahoo, and Deborah McGuinness. PROV-O: The PROV Ontology, 2013. W3C Recommendation. Consulted `http://www.w3.org/TR/2013/REC-prov-o-20130430/`.
20. J. W. Lloyd. *Foundations of Logic Programming (2nd edition)*. Springer-Verlag, 1987.
21. Robert M. MacGregor and In-Young Ko. Representing Contextualized Data using Semantic Web Tools. In *First International Workshop on Practical and Scalable Semantic Systems (PSSS-2003)*, 2003.
22. John McCarthy. Notes on Formalizing Context. In *13th International Joint Conference on Artificial Intelligence (IJCAI-2013)*, pages 555–562, 1993.
23. John Mylopoulos, Alexander Borgida, Matthias Jarke, and Manolis Koubarakis. Telos: Representing Knowledge about Information Systems. *ACM Transactions on Information Systems*, 8(4):325–362, October 1990.
24. Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of SPARQL. *ACM Transactions on Database Systems*, 34(3), 2009. Article No. 16.
25. E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Recommendation, 15 January 2008. Available at `http://www.w3.org/TR/rdf-sparql-query/`.
26. Manos Theodorakis, Anastasia Analyti, Panos Constantopoulos, and Nicolas Spyratos. Querying Contextualized Information Bases. In *24th Intern. Conference on Information and Communication Technologies and Programming (ICT&P-1999)*, 1999.

Appendix A: Table of Symbols

List of Symbols	
Symbol	*Description*
BN	the set of blank nodes
IRI	the set of IRI references
$\mathcal{LIT}$	the set of literals
Nam	the set of names
D	$\{rdf{:}langString, xsd{:}string\}$
c-BN	the set of contextual blank nodes
c-IRI	the set of contextual IRI references
c-$\mathcal{LIT}$	the set of contextual literals
c-Nam	the set of contextual names
$CNAM$	the set of context names
nam_c	the name of a context c
G_c	the c-RDF graph of a context c
ref_c	the references of a context c
V_c	the vocabulary of context c
CXT	the set of contexts
$\mathcal{V}_{RDF}^{\#n}$	$\mathcal{V}_{RDF} - \{rdf{:}\_i \mid i > n\}$
V_{CS}	$\cup\{V_c \mid c \in CS\} \bigcup \{[nil].u \mid u \in \mathcal{V}_{RDF}^{\#n_{CS}} \cup \mathcal{V}_{RDFS} \cup D\}$
$\mathcal{M}(CS)$	the models of contextual structure CS
$cl(CS)$	the closure of contextual structure CS
nam_{CS}	$\{nam_c \mid c \in CS\}$

Table 2. Symbols and Description

Appendix B: Proofs

In this Appendix, we provide the proof of the Propositions that appear in the main paper.

Proposition 1 Let G, G' be RDF graphs such that $max(\{i \in I\!N \mid \texttt{rdf:\_i}$ appears in $G'\}) \leq max(\{i \in I\!N \mid \texttt{rdf:\_i}$ appears in $G)\}$. Consider now the contextual structures $CS = \{\langle dummy, G, m\rangle\}$ and $CS' = \{\langle dummy, G', m\rangle\}$, where m is the empty mapping. Then, $G \models^{RDFS} G'$ iff $CS \models CS'$.

Proof: First, we provide a few definitions. Let G be an RDF graph. We denote by BN_G the set of blank nodes appearing in G. We denote by V_G the set of IRIS and literals appearing in G. We define $VBN_G = V_G \cup BN_G$. We define by n_G to be the maximum of 1 and the largest i such that $rdf{:}\_i$ appears in G.

We denote by Π_G, the following set of rules:

Simple Interpretation Rules

(1) $H(?z, \texttt{type}, \texttt{Property}) \leftarrow H(?x, ?z, ?y)$.
(2) $H(?x, \texttt{type}, \texttt{Resource}) \leftarrow H(?x, ?z, ?y)$.
(3) $H(?y, \texttt{type}, \texttt{Resource}) \leftarrow H(?x, ?z, ?y)$.

RDFS Interpretation Rules

(1) $H(?z, \texttt{type}, ?y) \leftarrow H(?x, \texttt{domain}, ?y), H(?z, ?x, ?w)$.
(2) $H(?w, \texttt{type}, ?y) \leftarrow H(?x, \texttt{range}, ?y), H(?z, ?x, ??w)$.
(3) $H(?x, \texttt{subClassOf}, \texttt{Resource}) \leftarrow H(?x, \texttt{type}, \texttt{Class})$.

(4) $H(?z, \texttt{type}, ?y) \leftarrow H(?x, \texttt{subClassOf}, ?y), H(?z, , \texttt{type}, ?x)$.

(5) $H(?x, \texttt{subClassOf}, ?x) \leftarrow H(?x, \texttt{type}, \texttt{Class})$.
(6) $H(?z, \texttt{subClassOf}, ?y) \leftarrow H(?x, \texttt{subClassOf}, ?y), H(?z, \texttt{subClassOf}, ?x)$.

(7) $H(?z_1, ?y, ?z_2) \leftarrow H(?x, \texttt{subPropertyOf}, ?y), H(?z_1, ?x, ?z_2)$.

(8) $H(?x, \texttt{subPropertyOf}, ?x) \leftarrow H(?x, \texttt{type}, \texttt{Property})$.
(9) $H(?z, \texttt{subPropertyOf}, ?y) \leftarrow H(?x, \texttt{subPropertyOf}, ?y), H(?z, \texttt{subPropertyOf}, ?x)$.

(10) $H(?x, \texttt{subClassOf}, \texttt{Literal}) \leftarrow H(?x, \texttt{type}, \texttt{Datatype})$.
(11) $H(?x, \texttt{subPropertyOf}, \texttt{member}) \leftarrow H(?x, \texttt{type}, \texttt{ContainerMembershipProperty})$.

For $d \in D$:
(12) $H(d, \texttt{type}, \texttt{Datatype}) \leftarrow \texttt{true}$.

For each "s"^^$d \in V_G$, where $d \in D$:
(13) $H($"s"^^$d, \texttt{type}, d) \leftarrow \texttt{true}$.

For each $E \in V_G$, where E is not a language-tagged string:
(14) $\texttt{false} \leftarrow H(E, \texttt{type}, \texttt{rdf:langString})$.

For each $s \in V_G$, where s is not a well-typed string:
(15) $\texttt{false} \leftarrow H(s, \texttt{type}, \texttt{xsd:string})$.

For each RDF and RDFS axiomatic triple (s, p, o) s.t. $p, s, o \in V_{RDF}^{\#n_G} \cup \mathcal{V}_{RDFS}$:
(16) $H(s, p, o) \leftarrow \texttt{true}$.

RDF graph rules
For each $(s, p, o) \in G$:
(1) $H(s, p, o) \leftarrow \texttt{true}$.

These, except inference rules (14) and (15), represent the RDFS inference rules as provided in RDFS 1.1 model theory document [14].

$\Rightarrow$) Assume $G \models^{RDFS} G'$. We will show that $CS \models CS'$. If $\texttt{false} \in T^{\uparrow\omega}_{\Pi_G}$ then $\texttt{false} \in T^{\uparrow\omega}_{P_{CS}}(\emptyset)$ and, thus according to Proposition 2, GS has no c-RDFS interpretation. Assume that $\texttt{false} \notin T^{\uparrow\omega}_{\Pi_G}$. Let $closure(G) = \{(s, p, o) \mid H(s, p, o) \in T^{\uparrow\omega}_{\Pi_G}\}$. According to proposition in [14], since $G \models^{RDFS} G'$, it holds that there is a mapping μ from $BN_{G'}$ to VBN_G such that $\mu(G') \subseteq closure(G)$. Let c be the unique context in CS and let c' be the unique context in CS'. It follows that there is a mapping μ' from $\texttt{c-}BN_{c'}$ to $\texttt{c-}VBN_{CS}$ such that $\mu'(c') \subseteq G_{cl(c,CS)}$. Therefore, based on Proposition 3, it follows that $CS \models CS'$.

$\Leftarrow$) Assume $CS \models CS'$. We will show that $G \models^{RDFS} G'$. If $\texttt{false} \in T_{P^{\uparrow\omega}_{CS}}$ then according to Proposition 2, GS has no c-RDFS interpretation. It follows that $\texttt{false} \in T^{\uparrow\omega}_{\Pi_G}$ and thus, G has no RDFS interpretation. We assume that $\texttt{false} \notin T_{P^{\uparrow\omega}_{CS}}$. Let c be the unique context in CS and let c' be the unique context in CS'. Since $CS \models CS'$, it follows that there is a mapping μ from $\texttt{c-}BN_{c'}$ to $\texttt{c-}VBN_{CS}$ such that $\mu(c') \subseteq G_{cl(c,CS)}$. Therefore, there is a mapping μ' from $BN_{G'}$ to VBN_G such that $\mu'(G') \subseteq closure(G)$. Then, according to proposition in [14], $G \models^{RDFS} G'$. □

Proposition 2 Let CS be a contextual structure. It holds that $\mathcal{M}(CS) = \emptyset$ iff $\texttt{false} \in T^{\uparrow\omega}_{P_{CS}}(\emptyset)$.

Proof:
$\Rightarrow$) Assume that $\texttt{false} \notin T^{\uparrow\omega}_{P_{CS}}(\emptyset)$. We will construct a c-RDFS interpretation of the contextual structure CS that satisfies CS. Let $\mathsf{I} = \{I_c \mid c \in CS\}$, where I_c is a c-simple interpretation of the c-vocabulary V_{CS} defined after a few definitions. Let $c \in CS$. Let $W_c = \{([u_s].s, [u_p].p, [u_o].o) \mid H(nam_c, u_s, s, u_p, p, u_o, o) \in T^{\uparrow\omega}_{P_{CS}}(\emptyset)\}$. For every contextual language-tagged string $[nil].E \in V_{CS}$ with lexical form s and language tag t and $[nil].\langle s, t'\rangle$, where t' is t converted to lower case using US-ASCII rules, we define $sur([nil].\langle s, t'\rangle = [nil].E$. For every contextual well-typed string $[nil].\text{“}s\text{”}\hat{}\hat{}xsd{:}string$, we define $sur([nil].\text{“}s\text{”}) = [nil].\text{“}s\text{”}\hat{}\hat{}xsd{:}string$. Note that there are not contextual not well-typed strings in V_{CS}, since we have assumed that $\texttt{false} \notin T^{\uparrow\omega}_{P_{CS}}(\emptyset)$. We extend sur to be the identity mapping on contextual IRIs, contextual blank nodes and all other contextual literals appearing in W_c.

Let:

- I_{c_V} be the identity mapping on the contextual IRIs in V_{CS}.
- We define Res_{I_c} to be the set of all items appearing in W_c after replacing contextual language-tagged strings and contextual well-typed strings s by $sur^{-1}(s)$.
- We define $Prop_{I_c} = \{p \mid (p, \texttt{type}, \texttt{Property}) \in W_c\}$.
- If $p \in Prop_{I_c}$ then $PT_{I_c}(p) = \{\langle s, o\rangle \mid (sur(s), sur(p), sur(o)) \in W_c\}$.
- We define a partial mapping IL_{I_c} from the contextual literals in V_{CS} to Res_{I_c} as follows: If $[nil].E \in V_{CS}$ is a contextual language-tagged string with lexical form s and language tag t then $IL_{I_c}([nil].E) = [nil].\langle s, t'\rangle$, where t' is t converted to

lower case using US-ASCII rules. If $[nil].\text{“}s\text{”}\hat{}\hat{}xsd{:}string \in V_{CS}$ is a contextual well-typed string, we define $IL_{I_c}([nil].\text{“}s\text{”}\hat{}\hat{}xsd{:}string) = [nil].\text{“}s\text{”}\hat{}\hat{}xsd{:}string$. For the rest of the contextual literals in V_{CS}, IL_{I_c} is undefined.

– We define $ref_{I_c} = ref_c$.

It is easy to see that $\mathsf{I} = \{I_c \mid c \in CS\}$ is a c-RDFS interpretation of the contextual structure CS. For example, we will show that the c-RDFS interpretation condition 3 of Definition 10 holds. Assume that $\langle x, y\rangle \in PT_{I_c}(\texttt{domain})$ and $\langle z, w\rangle \in PT_{I_c}(x)$. Then $(sur(x), \texttt{domain}, sur(y)) \in W_c$ and $(sur(z), sur(x), sur(w)) \in W_c$. It follows from the c-RDFS interpretation rule (1) that $(sur(z), \texttt{type}, sur(y)) \in W_c$. Thus, $\langle z, y\rangle \in PT_{I_c}(\texttt{type})$. Therefore, $z \in CT_{I_c}(y)$.

We will now show that $\mathsf{I} \models CS$. Let $c \in CS$ and $t = ([u].s, [u'].p, [u''].o) \in compl\_graph(c)$. Let v be a valuation that maps contextual blank nodes to themselves. Since $([u].s, [u'].p, [u''].o) \in W_c$, it holds that $\langle [I_c + v]([u].s), [I_c + v]([u''].o)\rangle \in PT_{I_c}(I_c([u'].p))\rangle$. Therefore, $I_c, v \models t$. Since $ref_{I_c} = ref_c$, it holds that $I, v \models c$. Therefore, $\mathsf{I} \models CS$.

$\Leftarrow$) Assume that $\mathsf{I} = \{I_c \mid c \in CS\} \in \mathcal{M}(CS)$. We will show that $\texttt{false} \notin T^{\uparrow\omega}_{P_{CS}}(\emptyset)$. Let $W_c = \{([u_s].s, [u_p].p, [u_o].o) \mid H(nam_c,\ u_s, s, u_p, p, u_o, o) \in T^{\uparrow\omega}_{P_{CS}}(\emptyset)\} \cup \{\texttt{false} \mid \texttt{false} \in T^{\uparrow\omega}_{P_{CS}}(\emptyset)\}$. Additionally, let $W^n_c = \{([u_s].s, [u_p].p, [u_o].o) \mid H(nam_c,\ u_s, s,\ u_p, p, u_o, o) \in T^{\uparrow n}_{P_{CS}}(\emptyset)\} \cup \{\texttt{false} \mid \texttt{false} \in T^{\uparrow n}_{P_{CS}}(\emptyset)\}$. Since $\mathsf{I} \models CS$, it holds that there is a valuation v such that for each $c \in CS$, $I_c, v \models c$. We will show that $I_c, v \models t$, for each $t \in W_c$ and $\texttt{false} \notin W_c$, where $c \in CS$, based on induction. Since $I_c, v \models c$, it holds that $I_c, v \models t$, for each $t \in compl\_graph(c)$. For each RDF and RDFS axiomatic triple (s, p, o) s.t. $p, s, o \in V^{\#n_{CS}}_{RDF} \cup \mathcal{V}_{RDFS}$, it holds that $I_c, v \models ([nil].s, [nil].p, [nil].o)$. For each $d \in D$, it holds that $I_c, v \models ([nil].d, [nil].\texttt{type}, [nil].\texttt{Datatype})$ due to c-RDFS interpretation condition 16 of Definition 10. Further, for each $[nil].\text{“}s\text{”}\hat{}\hat{}d \in V_{CS}$, it holds that $I_c, v \models ([nil].\text{“}s\text{”}\hat{}\hat{}d, [nil].\texttt{type}, [nil].d)$ due to c-RDFS interpretation conditions 19 and 20 of Definition 10. Note that if $d = xsd{:}string$ then typed literal $\text{“}s'\text{”}\hat{}\hat{}d$ is always well-typed since $\mathcal{M}(CS) \neq \emptyset$. Therefore, $I_c, v \models t$, for each $t \in W^0_c$, and $\texttt{false} \notin W^0_c$, where $c \in CS$.

Assumption: Assume that $I_c, v \models t$, for each $t \in W^n_c$, and $\texttt{false} \notin W^n_c$, where $c \in CS$ and $n \in \mathbb{N}$.

It is easy to see that $I_c, v \models t$, for each $t \in W^{n+1}_c$. For example, consider the c-RDFS Interpretation Rule (1) and assume that $t_1 = ([u_x].x, [nil].\texttt{domain}, [u_y].y) \in W^n_c$ and $t_2 = ([u_z].z, [u_x].x, [u_w].w) \in W^n_c$ then $t_3 = (u_z].z, [nil].\texttt{type}, [u_y].y) \in W^{n+1}_c$. Consider that $I_c, v \models t_1$ and $I_c, v \models t_2$ then $I_c, v \models t_3$, due to c-RDFS interpretation condition 3 of Definition 10. We will show that $\texttt{false} \notin W^{n+1}_c$. Assume that $\texttt{false} \in W^{n+1}_c$. Then, either (i) there is $[nil].E \in V_{CS}$, where E is not a language-tagged string such that $([nil].E, [nil].\texttt{type}, [nil].\texttt{rdf:langString}) \in W^n_c$, or (ii) there is $[nil].s \in V_{CS}$, where s is not a well-typed string such that $([nil].s, [nil].\texttt{type}, [nil].\texttt{xsd:string}) \in W^n_c$. However, this is impossible, since $I_c, v \models$

t, for each $t \in W_c^n$, and according to c-RDFS interpretation conditions 19 and 20 of Definition 10.

Therefore, $I_c, v \models t$, for each $t \in W_c$, and $\texttt{false} \notin W_c$, where $c \in CS$. Therefore, $\texttt{false} \notin T_{P_{CS}}^{\uparrow\omega}(\emptyset)$. □

Proposition 3 Let CS, CS' be two contextual structures such that $nam_{CS'} \subseteq nam_{CS}$ and $n_{CS'} \leq n_{CS}$. Let m be a mapping from CS' to CS such that if $m(c) = c'$, it holds that $nam_c = nam_{c'}$. Then, $CS \models CS'$ iff (i) there is a mapping μ from $\bigcup_{c \in CS'}$ c-BN_c to c-VBN_{CS} such that $\mu(c) \subseteq G_{cl(m(c),CS)}$, for all $c \in CS'$, and (ii) if $ref_c(u) = u'$, for $c \in CS'$, then $ref_{m(c)}(u) = u'$.

Proof:
$\Rightarrow$) Assume that $CS \models CS'$, we will first show that there is a mapping μ from $\bigcup_{c \in CS'}$ c-BN_c to c-VBN_{CS} such that $\mu(c) \subseteq G_{cl(m(c),CS)}$. We construct a c-RDFS interpretation $\mathsf{I} = \{I_c \mid c \in CS\}$ of the contextual structure CS that satisfies CS, as in the proof of Proposition 2 ($\Rightarrow$).

Since $CS \models CS'$ and $\mathsf{I} \models CS$, it follows that $\mathsf{I}' \models CS'$, where $\mathsf{I}' = \{I_{m(c)} \mid c \in CS'\}$. Therefore, there is a valuation v' such that $I_{m(c)}, v' \models c$, for every $c \in CS'$. Thus, if $t = ([u].s, [u'].p, [u''].o) \in compl\_graph(c)$, $c \in CS'$ then it holds that $\langle [I_{m(c)} + v']([u].s), [I_{m(c)} + v']([u''].o)\rangle \in PT_{I_{m(c)}}(I_{m(c)}([u'].p))\rangle$. Thus, $(sur([I_{m(c)} + v']([u].s)), [u'].p, sur([I_{m(c)} + v']([u''].o)) \in W_{m(c)}$. Considering a mapping $\mu = sur \circ v'$ from $\bigcup_{c \in CS'}$ c-BN_c to c-VBN_{CS}, it holds that $\mu(c) \subseteq G_{cl(m(c),CS)}$. Further, since $I_{m(c)}, v' \models c$, for every $c \in CS'$, it holds that if $ref_c(u) = u'$ then $ref_{I_{m(c)}}(u) = u'$. Since $ref_{I_{m(c)}}(u) = ref_{m(c)}(u)$, item (ii) is satisfied.

$\Leftarrow$) Assume that there is a mapping μ from $\bigcup_{c \in CS'}$ c-BN_c to c-VBN_{CS} such that $\mu(c) \subseteq G_{cl(m(c),CS)}$ and (ii) if $ref_c(u) = u'$ then $ref_{m(c)}(u) = u'$. We will show that $CS \models CS'$. Let $\mathsf{I} \models CS$, , where $\mathsf{I} = \{I_c \mid c \in CS\}$, we will show that $\mathsf{I}' \models CS'$, where $\mathsf{I}' = \{I_{m(c)} \mid c \in CS'\}$. Let $W_c = \{([u_s].s, [u_p].p, [u_o].o) \mid H(nam_c, u_s, s, u_p, p, u_o, o) \in T_{P_{CS}}^{\uparrow\omega}(\emptyset)\}$. Additionally, let $W_c^n = \{([u_s].s, [u_p].p, [u_o].o) \mid H(nam_c, u_s, s, u_p, p, u_o, o) \in T_{P_{CS}}^{\uparrow n}(\emptyset)\}$. Since $\mathsf{I} \models CS$, it holds that there is a valuation v such that for each $c \in CS$, $I_c, v \models c$. We will show that $I_c, v \models t$, for each $t \in W_c$, where $c \in CS$, based on induction. Since $I_c, v \models c$, it holds that $I_c, v \models t$, for each $t \in compl\_graph(c)$. For each RDF and RDFS axiomatic triple (s, p, o) s.t. $p, s, o \in V_{RDF}^{\#n_{CS}} \cup \mathcal{V}_{RDFS}$, it holds that $I_c, v \models ([nil].s, [nil].p, [nil].o)$. For each $d \in D$, it holds that $I_c, v \models ([nil].d, [nil].\texttt{type}, [nil].\texttt{Datatype})$ due to c-RDFS interpretation condition 16 of Definition 10. Further, for each $[nil].\text{“}s\text{”}\hat{}\hat{}d \in V_{CS}$, it holds that $I_c, v \models ([nil].\text{“}s\text{”}\hat{}\hat{}d, [nil].\texttt{type}, [nil].d)$ due to c-RDFS interpretation conditions 19 and 20 of Definition 10. Note that if $d = xsd{:}string$ then typed literal $\text{“}s\text{”}\hat{}\hat{}d$ is always well-typed since it is assumed that $\texttt{false} \notin T_{P_{CS}}^{\uparrow\omega}(\emptyset)$. Therefore, $I_c, v \models t$, for each $t \in W_c^0$, where $c \in CS$.

Assumption: Assume that $I_c, v \models t$, for each $t \in W_c^n$, where $c \in CS$ and $n \in \mathbb{N}$.

It is easy to see that $I_c, v \models t$, for each $t \in W_c^{n+1}$. For example, consider the c-RDFS Interpretation Rule (1) and assume that $t_1 = ([u_x].x, [nil].\texttt{domain}, [u_y].y) \in W_c^n$ and $t_2 = ([u_z].z, [u_x].x, [u_w].w) \in W_c^n$ then $t_3 = (u_z].z, [nil].\texttt{type}, [u_y].y) \in$

W_c^{n+1}. Consider that $I_c, v \models t_1$ and $I_c, v \models t_2$ then $I_c, v \models t_3$, due to c-RDFS interpretation condition 3 of Definition 10.

Therefore, $I_c, v \models t$, for each $t \in W_c$, where $c \in CS$. Let $([c_s].s, [c_p].p, [c_o].o) \in compl\_graph(c)$, where $c \in CS'$. Extend μ such that μ on c-$IRI \cup$c-$\mathcal{LIT}$ is the identity function. Then, $(\mu([u_s].s), \mu([u_p].p), \mu([u_o].o)) \in G_{cl(m(c),CS)}$. Obviously, $I_{m(c)}, v \models (\mu([u_s].s), \mu([u_p].p), \mu([u_o].o))$. Define valuation v' such that (i) $v'(x) = v \circ \mu(x)$, if $\mu(x) \in$ c-BN, for x in $\bigcup_{c \in CS'}$ c-BN_c and (ii) $v'(x) = \mu(x)$, if $\mu(x) \in$ c-$IRI \cup$c-$\mathcal{LIT}$, for x in $\bigcup_{c \in CS'}$ c-BN_c. Then, $I_{m(c)}, v' \models ([u_s].s, [u_p].p, [u_o].o)$. Additionally, from item (ii), we have that if $ref_c(u) = u'$, for $c \in CS'$, then $ref_{m(c)}(u) = u'$. It holds that $ref_{I_{m(c)}}(u) = ref_{m(c)}(u)$. Thus, $I_{m(c)}, v' \models c$, for each $c \in CS'$. Therefore, $\mathsf{I}' \models CS'$. □

Proposition 4 Let CS, CS' be contextual structures such that $n_{CS'} \leq n_{CS}$. Checking if $CS \models CS'$ has NP-complete time complexity.

Proof: In [12] (Theorem 2.10), it is proved that deciding RDFS entailment between two RDF graphs is NP-complete. Therefore, based on Proposition 1, deciding if $CS \models CS'$ is an NP-hard problem. Now guess a mapping μ from $\bigcup_{c \in CS'}$ c-BN_c to c-VBN_{CS} and check if $nam_{CS'} \subseteq nam_{CS}$. If this holds, let m be a mapping from CS' to CS such that if $m(c) = c'$, it holds that $nam_c = nam_{c'}$. Now check (i) if $\mu(c) \subseteq G_{cl(m(c),CS)}$, for all $c \in CS'$, and (ii) if $ref_c(u) = u'$, for $c \in CS'$, then $ref_{m(c)}(u) = u'$. All checks can be done in polynomial time and if they hold it follows, based on Proposition 3, that $CS \models CS'$. Therefore, checking if $CS \models CS'$ can be done in NP-time. □

Proposition 5 Let CS and CS' be contextual structures such that $n_{CS'} \leq n_{CS}$. It holds that:
$CS \models CS'$ iff $cl(CS) \models CS'$.

Proof:
⇒) Let $CS \models CS'$. Then, $nam_{CS'} \subseteq nam_{CS}$. Let m be a mapping from CS' to CS such that if $m(c) = c'$, it holds that $nam_c = nam_{c'}$. Based on Proposition 3, (i) there is a mapping μ from $\bigcup_{c \in CS'}$ c-BN_c to c-VBN_{CS} such that $\mu(c) \subseteq G_{cl(m(c),CS)}$, for all $c \in CS'$ and (ii) if $ref_c(u) = u'$, for $c \in CS'$, then $ref_{m(c)}(u) = u'$. It holds that $nam_{CS'} \subseteq nam_{cl(CS)}$ and $n_{CS'} \leq n_{cl(CS)}$. Let m' be a mapping from CS' to $cl(CS)$ such that if $m'(c) = c'$, it holds that $nam_c = nam_{c'}$. It follows that (i) there is a mapping μ' from $\bigcup_{c \in CS'}$ c-BN_c to c-$VBN_{cl(CS)}$ such that $\mu(c) \subseteq G_{cl(m(c),cl(CS))}$, for all $c \in CS'$, and (ii) if $ref_c(u) = u'$, for $c \in CS'$, then $ref_{m'(c)}(u) = u'$. Therefore, based on Proposition 3, it follows that $cl(CS) \models CS'$.

⇐) Let $cl(CS) \models CS'$. Then, $nam_{CS'} \subseteq nam_{cl(CS)}$. Let m be a mapping from CS' to $cl(CS)$ such that if $m(c) = c'$, it holds that $nam_c = nam_{c'}$. Based on Proposition 3, (i) there is a mapping μ from $\bigcup_{c \in CS'}$ c-BN_c to c-$VBN_{cl(CS)}$ such that $\mu(c) \subseteq G_{cl(m(c),cl(CS))}$, for all $c \in CS'$ and (ii) if $ref_c(u) = u'$, for $c \in CS'$, then $ref_{m(c)}(u) = u'$. It holds that $nam_{CS'} \subseteq nam_{CS}$ and $n_{CS'} \leq n_{CS}$. Let m' be a mapping from CS' to CS such that if $m'(c) = c'$, it holds that $nam_c = nam_{c'}$. It follows that (i) there is a mapping μ' from $\bigcup_{c \in CS'}$ c-BN_c to c-VBN_{CS} such that $\mu(c) \subseteq G_{cl(m(c),CS)}$, for

all $c \in CS'$, and (ii) if $ref_c(u) = u'$, for $c \in CS'$, then $ref_{m'(c)}(u) = u'$. Therefore, based on Proposition 3, it follows that $CS \models CS'$. □

Proposition 6 Let CS be a contextual structure and let $c \in CS$. Let t be a c-RDF triple consisting of contextual names that are not `rdf:_i` terms such that $t \notin G_{cl(c,CS)}$. Let c' be a new context such that $c' = \langle nam_c, G_{cl(c,CS)} \cup \{t\}, ref_c \rangle$. Let $CS' = (cl(CS) \setminus cl(c, CS)) \cup \{c'\}$. Then, it holds that $CS \not\models CS'$.

Proof: It holds that $nam_{CS'} \subseteq nam_{CS}$ and $n_{CS'} \leq n_{CS}$. Let m be a mapping from CS' to CS such that if $m(c) = c''$, it holds that $nam_c = nam_{c''}$. However, note that it is not possible to find mapping μ from $\bigcup_{c \in CS'}$ c-BN_c to c-VBN_{CS} such that $\mu(c') \subseteq G_{cl(m(c'),CS)}$. Therefore, based on Proposition 3, it follows that $CS \not\models CS'$. □

Proposition 7 Let CS be a contextual structure and $c_1, c_2 \in CS$. The complexity of the algorithm *context_incorpor*(c_1, c_2, CS) is in $O(n_{c_1} * S_{max})$, where n_{c_1} is the number of subcontexts of c_1 and S_{max} is the size of the largest context c_1, c_2 and their subcontexts (the size is in terms of the number of c-RDF triples and the number of references of the contexts).

Proof: The proof is straightforward since the algorithm *context_incorpor*(c_1, c_2, CS) visits all subcontexts of c_1 until leaf subcontexts are reached and at each iteration the full contents of a subcontext of c_1 with a corresponding subcontext of c_2 are checked. □