NdFluents: An Ontology for Annotated Statements with Inference Preservation

José M. Giménez-García[1(✉)], Antoine Zimmermann[2], and Pierre Maret[1]

[1] Université de Lyon, CNRS, UMR 5516,
Laboratoire Hubert-Curien, Saint-Étienne, France
jose.gimenez.garcia@univ-st-etienne.fr
[2] Université de Lyon, MINES Saint-Étienne, CNRS, Laboratoire Hubert Curien
UMR 5516, 42023 Saint-Étienne, France

Abstract. RDF provides the means to publish, link, and consume heterogeneous information on the Web of Data, whereas OWL allows the construction of ontologies and inference of new information that is implicit in the data. Annotating RDF data with additional information, such as provenance, trustworthiness, or temporal validity is becoming more and more important in recent times; however, it is possible to natively represent only binary (or dyadic) relations between entities in RDF and OWL. While there are some approaches to represent metadata on RDF, they lose most of the reasoning power of OWL. In this paper we present an extension of Welty and Fikes' 4dFluents ontology—on associating temporal validity to statements—to any number of dimensions, provide guidelines and design patterns to implement it on actual data, and compare its reasoning power with alternative representations.

Keywords: Annotations · Contexts · Metadata · Ontologies · OWL · RDF · Reasoning · Reification

1 Introduction

The Resource Description Framework (RDF) represents statements as triples that typically match phrases with a subject, a verb and a complement. However, it is often the case that more complex information has to be encoded, such as qualifying a statement with its origin, its validity within a time frame, its degree of certainty, and so on. In this case, one may have to represent statements about a statement. We describe this as an annotated statement. However, with the RDF model it is only possible to represent binary (or dyadic) relations between subject and object [10]. In order to represent additional data about statements it is usually needed to use external annotations, extend either the data model [1] or the semantics of RDF [4,11], or use design patterns to represent that information [2,12].

On the other hand, RDF Schema (RDFS) and the Web Ontology Language (OWL) add formal semantics to RDF, making it possible to infer new statements

E. Blomqvist et al. (Eds.): ESWC 2017, Part I, LNCS 10249, pp. 638–654, 2017.
DOI: 10.1007/978-3-319-58068-5_39

from pre-existing knowledge. However, when data is annotated using the previous approaches, the inferences in the original dataset are no longer possible, or the new inferred data is missing part of the annotations. For instance, OWL allows to define a relation between two resources as transitive. In that case, if a resource A is related to another resource Busing that property, and B is in turn related with another resource C with the same property, then it is inferred that A and C are also related. This inference is not preserved when using Reification, a classic approach to reference a triple and annotate it with metadata, that removes the original triple and replaces it with four new triples to identify the statement and describe the position of each element of the original triple.

Along these lines, Welty and Fikes [16] proposed an ontology for representing temporally changing information using a perdurantist view, where statements are asserted over temporal slices of entities, retaining most reasoning capabilities. This approach can be generalized to annotate data not only with temporal information, but with information from any dimension [15]. However, modeling several context dimensions for a statement is not straightforward and presents some challenges. In this work, we propose a generalization of Welty and Fikes model in the form of a generic ontology that can be extended to implement any number of concrete metadata dimensions, while preserving reasoning capacity relative to each dimension.

The rest of the paper is structured as follows: Sect. 2 presents the 4dFluents ontology for annotating statements with temporal data; Sect. 3 introduces NdFluents, the generalization of 4dFluents to annotate statements with any number of context dimensions; Sect. 4 describes three design patterns that can be used to model a combination of context dimensions; Sect. 5 discusses issues and possible solutions when representing metadata with NdFluents; Sect. 6 compares the reasoning capabilities of NdFluents with other current approaches to represent metadata about statements in RDF; Sect. 7 portrays related work; finally, we present some conclusions in Sect. 8.

2 Welty and Fikes' 4dFluents Ontology

Welty and Fikes [16] address the problem of representing *fluents*, *i.e.*, relations that hold within a certain time interval and not in others. They address the issue from the perspective of diachronic identity (that is, how an entity looks to be different at different times), showcasing the two ways of tackling it:

- The *endurantist* (*3D*) view maintains a differentiation between *endurants*, entities that are present at all times during its whole existence, and *perdurants*, events affecting an entity during a definite period of time during the entity's existence.
- The *perdurantist* (*4D*) view argues that entities themselves have to be handled as perdurants, *i.e.*, temporal parts of a four dimensional meta-entity. Instead of making an assertion about some entities, such as *"Paris is the capital of France"*, one should make the assertion about their temporal parts: *"A temporal part of Paris (since 508 up to now) is the capital of a temporal part of France (since 508 up to now)"*.

Welty and Fikes adopt the perdurantist approach to create the *4dFluents* ontology, representing *entities at a time* and using them as resources for their statements. The 4dFluents ontology expressed in OWL2 Functional Syntax is shown in Ontology 1.

```
Prefix( 4d:=<http://www.example.com/4dFluents#> )
Ontology( <http://www.example.com/4dFluents>
        Declaration( Class( 4d:Interval ) )
        Declaration( Class( 4d:TemporalPart ) )
        DisjointClasses( 4d:Interval 4d:TemporalPart )

        Declaration( ObjectProperty( 4d:fluentProperty ) )
        ObjectPropertyDomain( 4d:fluentProperty 4d:TemporalPart )
        ObjectPropertyRange( 4d:fluentProperty 4d:TemporalPart )

        Declaration( ObjectProperty( 4d:temporalExtent ) )
        FunctionalObjectProperty( 4d:temporalExtent )
        ObjectPropertyDomain( 4d:temporalExtent 4d:TemporalPart )
        ObjectPropertyRange( 4d:temporalExtent 4d:Interval )

        Declaration( ObjectProperty( 4d:temporalPartOf ) )
        FunctionalObjectProperty( 4d:temporalPartOf )
        ObjectPropertyDomain( 4d:temporalPartOf 4d:TemporalPart )
        ObjectPropertyRange( 4d:temporalPartOf ObjectComplementOf( 4d:Interval ))
)
```

Ontology 1. 4dFluents ontology (from [16])

In order to use the ontology for describing fluents, one has to introduce axioms at the terminological level (TBox) as well as assertions in the knowledge base (ABox). For instance, if one wants to say that *"Paris is the capital of France"* since 508, the relation "capital of" has to be a subproperty of `fluentProperty` and new individuals have to be introduced for the temporal part of Paris and of France, as shown in Ontology 2.

```
Declaration( ObjectProperty( ex:capitalOf ) )
SubObjectPropertyOf( ex:capitalOf 4d:fluentProperty )
ClassAssertion( 4d:TermporalPart ex:Paris@508 )
ClassAssertion( 4d:TermporalPart ex:France@508 )
ClassAssertion( 4d:Interval ex:year508) )
ObjectPropertyAssertion( ex:capitalOf ex:Paris@508 ex:France@508 )
ObjectPropertyAssertion( 4d:temporalExtent ex:Paris@508 ex:year508 )
ObjectPropertyAssertion( 4d:temporalExtent ex:France@508 ex:year508 )
ObjectPropertyAssertion( 4d:temporalPartOf ex:Paris@508 ex:Paris )
ObjectPropertyAssertion( 4d:temporalPartOf ex:France@508 ex:France )
```

Ontology 2. Expressing a fact about a fluent entity with the 4dFluents ontology

In this way, temporal information can be represented with standard OWL semantics, preserving reasoning capabilities.

3 The NdFluents Ontology

A temporal part of an entity can be viewed as an individual context dimension of the entity. A similar approach can then be used to represent different dimensions, such as provenance or confidence. Continuing with our running example, if Wikipedia states that *"Paris is the capital of France"*, we can articulate that fact as *"Paris as defined by Wikipedia is the capital of France as defined by Wikipedia"*. Different context dimensions of an entity could then be combined if applicable, allowing the representation of complex information, such as: *"A temporal part of Paris as defined by Wikipedia is the capital of a temporal part of France as defined by Wikipedia"*.

We use this idea to extend the 4dFluents ontology for an arbitrary number of context dimensions in the *NdFluents* ontology. The ontology, shown in Ontology 3, and published in http://www.emse.fr/~zimmermann/ndfluents.html, is a generalization from temporal parts to contextual parts.

```
Prefix( nd:=<http://purl.org/NET/NdFluents#> )
Ontology( <http://purl.org/NET/NdFluents>
        Declaration( Class( nd:Context ) )
        Declaration( Class( nd:ContextualPart ) )
        DisjointClasses( nd:Context nd:ContextualPart )

        Declaration( ObjectProperty( nd:contextualProperty ) )
        ObjectPropertyDomain( nd:contextualProperty nd:ContextualPart )
        ObjectPropertyRange( nd:contextualProperty nd:ContextualPart )

        Declaration( ObjectProperty( nd:contextualExtent ) )
        ObjectPropertyDomain( nd:contextualExtent nd:ContextualPart )
        ObjectPropertyRange( nd:contextualExtent nd:Context )

        Declaration( ObjectProperty( nd:contextualPartOf ) )
        FunctionalObjectProperty( nd:contextualPartOf )
        ObjectPropertyDomain( nd:contextualPartOf nd:ContextualPart )
        ObjectPropertyRange( nd:contextualPartOf ObjectComplementOf( nd:Context ))
)
```

Ontology 3. The NdFluents ontology

Note that `FunctionalObjectProperty( nd:contextualExtent )` axiom is not present in the ontology. This axiom should appear if the ontology was a direct translation from temporal dimension to a generic context dimension, but it is no longer applicable when we have more than one dimension simultaneously.

The NdFluents ontology is meant to be implemented for different context dimensions in a modular way. In this sense, the 4dFluents ontology can be seen as a concrete implementation of NdFluents, as we show in Ontology 4. In Fig. 1a we show the representation of a statement with temporal annotations using this ontology. The non-dashed parts are equivalent to the original 4dFluents ontology, while the dashed parts correspond to the NdFluents extension. Other dimensions, such as provenance, can be modeled similarly to the temporal

dimension by replacing `TemporalPart` with `ProvenancePart`, `temporalExtent` with `provenanceExtent`, `Interval` with `Provenance`, and `temporalPartOf` with `provenancePartOf`. Additionally, an assertion like *"Paris is the capital of France, according to Wikipedia"* can be modeled following the same pattern as in Ontology 2, replacing the property and class names with their counterparts in the provenance dimension.

```
Prefix( nd:=<http://purl.org/NET/ndfluents#> )
Prefix( 4d:=<http://purl.org/NET/ndfluents/4dFluents#>)
Ontology( <http://www.example.com/4dFluentsV2>
        Import( <http://www.example.com/NdFluents> )

        Declaration( Class( 4d:Interval ) )
        SubClassOf( 4d:Interval nd:Context )
        Declaration( Class( 4d:TemporalPart ) )
        SubClassOf( 4d:TemporalPart nd:ContextualPart )

        Declaration( ObjectProperty( 4d:temporalExtent ) )
        SubObjectPropertyOf( 4d:temporalExtent nd:contextualExtent )
        ObjectPropertyDomain( 4d:temporalExtent 4d:TemporalPart )
        ObjectPropertyRange( 4d:temporalExtent 4d:Interval )

        Declaration( ObjectProperty( :temporalPartOf ) )
        SubObjectPropertyOf( 4d:temporalExtent nd:contextualPartOf )
        ObjectPropertyDomain( 4d:temporalPartOf 4d:TemporalPart )
)
```

Ontology 4. 4dFluents ontology as implementation of NdFluents

4 Design Patterns

An important scenario where NdFluents becomes relevant is when the necessity of combining two or more context dimensions arises, such as *"According to Wikipedia, Paris is the capital of France since 508"*. In this section we present three design patterns to combine different dimensions, along with added axioms that can be necessary depending on the modeling needs. Methodological support for choosing and implementing a design pattern can be found at Giménez-García et al. [3]

4.1 Contexts in Context

One possible model to represent information using different context dimensions is to relate a `ContextualPart` to another `ContextualPart`. This approach can be taken when the "first level" annotations are relevant facts of the knowledge base, and the intention is to state additional information about them. To be able to reason about different annotation levels of any entity, it is desirable for the

`contextualPartOf` property to be transitive, which can be achieved by adding the axiom of Ontology 5.

While data about different dimensions can be more fine-grained using this model, it also grows in complexity. For example, in Fig. 1b the statement `capitalOf` is related to the `ProvenancePart Paris@1.1`. This information is in no way related to the `TemporalPart Paris@1`. While we could have this statement duplicated in the example, this can become unfeasible when we start adding more contextual parts to the data. We believe that this pattern can be useful in some specific cases, but it is usually too cumbersome.

```
Prefix( nd:=<http://purl.org/NET/ndfluents#> )
Ontology( <http://purl.org/NET/ndfluents/transitivecontextualpartof>
      TransitiveObjectProperty( nd:contextualPartOf )
)
```

Ontology 5. Transitive axiom for NdFluents ontology

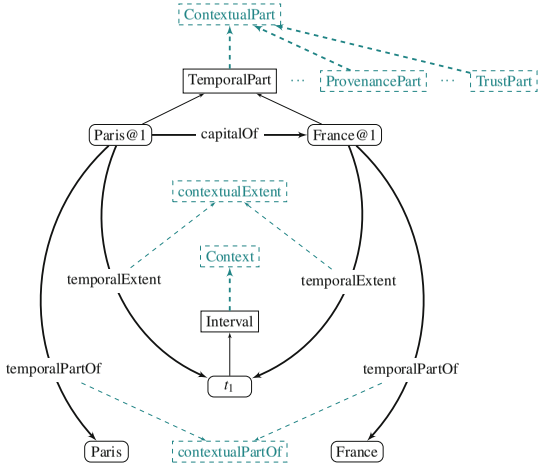
4.2 Use Multiple Contextual Extents on Each Contextual Part

A generic approach for representing entities with more than one context dimension is to have `ContextualParts` with more than one contextual extent. Using this model, only one `ContextualPart` is created for a combination of context dimensions. This `ContextualPart` is then related to all related contextual extents, as shown in Fig. 1c. This pattern is easier to model: Relating the `ContextualPart` with the context dimensions is straightforward. It also avoids ambiguity when modeling annotations related to more than one dimension, and reduces the number of resources in the ontology (*i.e.*, while the previous model needed one `ContextualPart` for each dimension involved, this approach only requires one). Note that `contextualPartOf` is a functional property, which means that there cannot be a `contextualPartOf` of more than one entity.

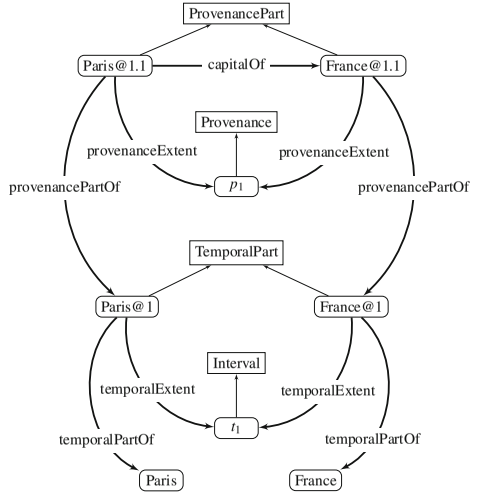
4.3 Combine Different Contexts on One Contextual Extent

Finally, a third possibility is to create compound `Contexts`, and enforce a limit of only one `Context` per `ContextualPart`. This model adds a layer of complexity to the previous approach, but it can be useful to require a specific combination of dimensions on a set of `ContextualParts`. This can be achieved by adding the axiom in Ontology 6.

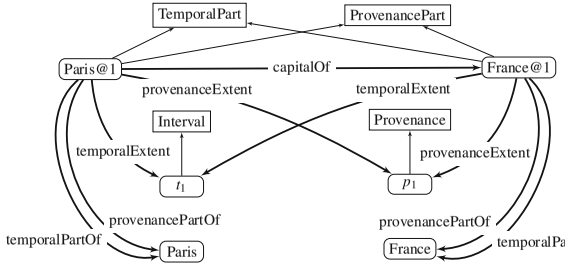
We show an example of this approach on Fig. 1d. Note that the combined classes and properties are subclasses and subproperties of the corresponding classes and properties of the two context dimensions they are combining (*e.g.*, `Temporal+ProvenancePart` is subclass of `TemporalPart` and `ProvenancePart`). As a result, querying and reasoning can be performed in an identical way as the previous approach.



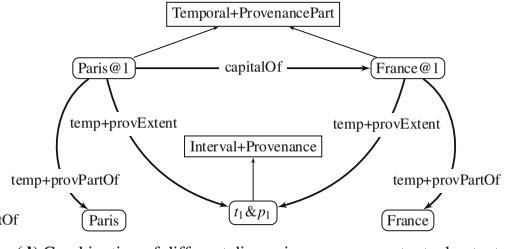
(a) Example of 4dFluents as implementation of NdFluents



(b) Contexts in context



(c) Multiple Contexts on each ContextualPart



(d) Combination of different dimensions on one contextual extent

Fig. 1. NdFluents ontology and design patterns

```
Prefix( nd:=<http://purl.org/NET/ndfluents#> )
Ontology( <http://purl.org/NET/ndfluents/functionalcontextualExtent>
        FunctionalObjectProperty( nd:contextualExtent )
)
```

Ontology 6. Functional contextual extents axiom for NdFluents ontology

5 Additional Considerations

In this section we discuss issues that may arise when modeling annotations using fluents, and possible approaches to deal with them if they exist. While the first one is common to the original 4dFluents ontology, the second is only relevant when dealing with more than one context dimension.

5.1 Dealing with Datatype Properties

The original 4dFluents ontology does not provide any information for modeling datatype properties. While there is nothing that prevents using regular datatype properties with `ContextualParts` of an entity, it may be desirable to declare explicit axioms for annotation properties to facilitate reasoning on that information. In that case, the statements of Ontology 7 need to be added to the NdFluents ontology. Figure 2 shows an example where a annotated property is used to state the population of Paris in a specific temporal interval. Note that it is also possible to create specific `contextualProperty` subproperties for different context dimensions (*i.e.*, `temporalProperty` for `TemporalPart`) for properties related to concrete context dimensions.

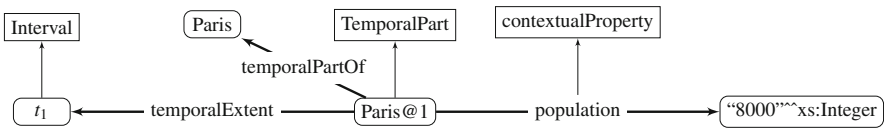


Fig. 2. Example of annotated datatype property

```
Prefix( nd:=<http://purl.org/NET/ndfluents#> )
Ontology( <http://purl.org/NET/ndfluents/annotatedDatatypeProperty>
        Declaration( DataProperty( nd:annotatedDatatypeProperty ) )
        DataPropertyDomain ( nd:annotatedDataProperty nd:
            ContextualPart )
)
```

Ontology 7. Datatype axioms for NdFluents ontology

5.2 Relations Between ContextualParts of Different Dimensions

The NdFluents ontology presented thus far allows the modeling of relations among different ContextualParts of different dimensions (*i.e.*, a TemporalPart of Paris could be the capital of a ProvenancePart of France). While this can be convenient for individual cases, it is often needed for an contextualProperty to be related to ContextualParts of the same dimension. In this case, it is necessary to add the appropriate axioms to the ontology. In Ontology 8. we show the needed axioms to include this restriction on the TemporalParts. Conversely, if there are datatype properties related to specific dimensions, axioms from Ontology 9 should be added.

```
Prefix( nd:=<http://purl.org/NET/ndfluents#> )
Prefix( 4d:=<http://purl.org/NET/ndfluents/4dFluents#>)
Ontology( <http://purl.org/NET/ndfluents/4dFluents/
    temporalpartrestriction>
       Declaration( ObjectProperty( 4d:fluentProperty ) )
       SubObjectPropertyOf( 4d:fluentProperty nd:contextualProperty )
       ObjectPropertyDomain( 4d:fluentProperty 4d:TemporalPart )
       ObjectPropertyRange( 4d:fluentProperty 4d:TemporalPart )
)
```

Ontology 8. Temporal restriction on object properties 4dFluents ontology

```
Prefix( nd:=<http://purl.org/NET/ndfluents#> )
Prefix( 4d:=<http://purl.org/NET/ndfluents/4dFluents#>)
Ontology( <http://purl.org/NET/ndfluents/4dFluents/
    temporalpartrestriction>
       Declaration( DataProperty( 4d:fluentDataTypeProperty ) )
       SubDataPropertyOf( 4d:fluentDataTypeProperty nd:
           contextualProperty )
       DataPropertyDomain( 4d:fluentProperty 4d:TemporalPart)
)
```

Ontology 9. Temporal restriction on datatype properties 4dFluents ontology

In a similar fashion, it is usually desirable that ContextualParts of the same dimension relate to the same Context. That is, if a Provenance Part of Paris relates to a ProvenancePart of France, their provenanceExtent properties should have the same ProvenancePart object. However, this restriction cannot be expressed in OWL. If needed, a rule language (such as SWRL [7] or RIF [8]) can be used for this purpose, but this case goes beyond the scope of this paper.

6 Reasoning with Annotated Data

In this section, we compare the reasoning capabilities of the NdFluents ontology with other approaches to annotate statements, namely RDF reification, N-ary relations, and singleton property. The interest is to know what RDFS and OWL entailments are preserved wrt the original unannotated data. For that, we need to formally define what annotations and entailment preservation mean. We assume that annotated statements can be described as a pair (G, A) where G is the graph corresponding to the statements that are annotated, and A denotes the annotations on G. The structure of A could be arbitrarily complex (e.g., containing dates, creator, provenance) but for the sake of this section and to simplify the presentation, we simply assume that the annotation structure is identified with an IRI. Thus, we approximate the notion of annotated statements with the concept of named graphs, i.e., pairs (n, G) where n is an IRI and G is an RDF graph. However, there is no standard way of reasoning with named graphs [19]. Our objective then is to compare approaches that convert annotated statements into RDF graphs. We name such approaches *RDF representation of annotated statements* and formalize it as follows.

Definition 1 (RDF representation of annotated statements). An *RDF representation of annotated statements* is a function f that maps annotated statements (in our simplified model, named graphs) (n, G) to an RDF graph $f(n, G)$.

For examples of this function, refer to Subsect. 6.1, where we describe four existing models to annotate statements and present their corresponding functions.

We want to assess to what extent each representation is preserving entailment with the notions of entailment preservation (when the entailment preserves also the annotations) and non-contextual entailment preservation (when only the original entailment is preserved) defined as follows.

Definition 2 (Entailment preservation). Let G_1 and G_2 be two RDF graphs such that $G_1 \models G_2$ and f be an RDF representation of annotated statements.[1] We say that f *preserves the entailment* between G_1 and G_2 iff for all annotation IRI n, $f(n, G_1) \models f(n, G_2)$.

Definition 3 (Non-Contextual Entailment preservation). Let G_1 and G_2 be two RDF graphs such that $G_1 \models G_2$ and f be an RDF representation of annotated statements.[1] We say that f *non-contextually preserves the entailment* between G_1 and G_2 iff for all annotation IRI n, $f(n, G_1) \models G_2$.

We generalize these notions to the case of entailment rules of the form $P(\mathbf{x}) \leftarrow Q(\mathbf{x}, \mathbf{y})$, where P and Q are graph patterns and $\mathbf{x}$, $\mathbf{y}$ are tuples of variables used in the patterns.

[1] This definition can apply to any entailment regime so that it is not necessary to specify what the relation $\models$ exactly is.

Definition 4 (Rule preservation). Let $R = P(\mathbf{x}) \leftarrow Q(\mathbf{x}, \mathbf{y})$ be a rule and f an RDF representation of annotated statements. We say that f *preserves the rule* R iff for all mappings μ from variables in $\mathbf{x}$ and $\mathbf{y}$ to RDF terms, $f(n, Q(\mu(x), \mu(y))) \models f(n, P(\mu(x)))$.

Definition 5 (Non-Contextual Rule preservation). Let $R = P(\mathbf{x}) \leftarrow Q(\mathbf{x}, \mathbf{y})$ be a rule and f an RDF representation of annotated statements. We say that f *non-contextually preserves the rule* R iff for all mappings μ from variables in $\mathbf{x}$ and $\mathbf{y}$ to RDF terms, $f(n, Q(\mu(x), \mu(y))) \models P(\mu(x))$.

For example, if we have an inference rule that allows us to infer that `(France, hasCapital,Paris)` from the triple `(Paris,capitalOf,France)`, and we have an representation of annotated statements for `(Paris,capitalOf,France), (508,now)`, rule preservation would allow us to infer `(France, hasCapital, Paris),(508,now)`, while non-contextual rule preservation would allow to infer `(France,hasCapital,Paris)` from the annotated triple. Note this kind of inferences function annotates triples of the condition but the conclusion is not annotated are not always desirable. This will be further explained in Subsect. 6.2.

In the following subsections we first present the *RDF representation of annotated statements* (see Definition 1) for the representation approaches, and then proceed to compare the rule preservation for each one of them.

6.1 RDF Representation Approaches

- **Reification**[2] is the standard W3C model to represent information about an statement, proposed in 2004. A triple is represented as an instance of `rdf:Statement`, that relates to the original triple with the properties `rdf:subject`, `rdf:predicate` and `rdf:object`. Then, a triple (s, p, o) is replaced by the following set: $\{(i$,`rdf:type`,`rdf:Statement`$), (i$,`rdf:subject`$, s), (i$,`rdf:predicate`$, p), (i$,`rdf:object`$, o)\}$, and annotations are related to i.
- **N-Ary relations** [12] were proposed in 2006 to represent relations between more than two individuals, or to describe the relation themselves. In this model, an individual is created to represent the relation, which can be used as the subject for new statements. Thus, a triple (s, p, o) is replaced by the following set: $\{(s, p'_1, r), (r, p'_2, o)\}$, and annotations are related to r.
- The **Singleton Property** [11] is a recent proposal to represent information about statements in RDF. A particular instance of the predicate is created for every triple. This instance is related to the original predicate by the `singletonPropertyOf` property. Then, each statement can be unequivocally referenced using its predicate for attaching additional information. Therefore, a triple (s, p, o) is replaced by the set: $\{(s, p', o)$, $(p'$,`sp:singletonPropertyOf`$, p)\}$, and annotations are related to p'.
- **NdFluents**, the approach presented in this paper, creates a contextualized individual for both subject and object (in case it is a URI or blank node) of the triple. The triple is the replaced by a new one that uses the contextualized individuals. These two new resources are related to the original individuals

[2] https://www.w3.org/TR/2004/REC-rdf-primer-20040210/#reification.

and with a Context, where the annotations are attached. Hence, the original triple (s, p, o) is replaced by the following set of triples $\{(s_c, p, o_c)$, $(s_c$,`nd:contextualPartOf`$, s)$, $(o_c$,`nd:contextualPartOf`$, o)$, $(s_c$,`nd:contextual Extent`$, c)$, $(o_c$,`nd: contextual Extent`$, c)\}$, where c is a function of the context. Annotations are related to c.

.

6.2 Comparison of Rule Preservation

For comparing how entailment is preserved in each of the 4 approaches presented in Sect. 6.1, we analyze which rules from the pD* fragment of OWL ter Horst [14] are preserved. This fragment is a modified subset of RDFS and OWL that can be expressed as a complete set of rules and is computationally feasible. For each rule, we check if is in accordance with *Rule Preservation* and *Non-Contextual Rule Preservation* (*i.e.*, for the former, if the inference rule holds when we apply the *RDF representation of annotated statements* function to both condition and conclusion; for the latter, if it holds when we apply the function only to the condition). It is important to note that the representation approaches are usually used to annotate data on relations between resources. For this reason, we decide to implement the representations on triples that do not include RDF, RDFS, or OWL vocabularies.

Table 1 shows the D* (modified RDFS) entailment rules and rule preservations for each one of the approaches, whereas Table 2 presents the same information for P entailments (modified subset of OWL). Note that we remove those rows where both condition and conclusion include only triples with RDF, RDFS, or OWL vocabularies. A P indicates that there is rule preservation for the corresponding approach, while a P_{NC} denotes non-contextual rule preservation. As mentioned in Sect. 6, it is worth noting that not all rule preservations are desirable. When the preserved rule entails new knowledge on the non-annotated graph, and the annotated triples are not universally true, then the inferences can lead to conclusions that do not conform with real-world knowledge. This

Table 1. Preserved D* entailments (P = Rule preservation, P_{NC} = Non-contextual rule preservation, ! = Risk of undesirable inference)

Rule	Condition	Constraint	Conclusion	Reif.	N-Ary	S.P.	NdF
lg	$v\ p\ l$	$l \in L$	$v\ p\ b_l$	P	P	P	P
gl	$v\ p\ b_l$	$l \in L$	$v\ p\ l$	P	P	P	P
rdf1	$v\ p\ w$		p `type Property`			P	P
rdf2-D	$v\ p\ l$	$l = (s,a) \in L_D^+$	b_l `type` a	P	P	P	P
rdfs1	$v\ p\ l$	$l \in L_p$	b_l `type Literal`	P	P	P	P
rdfs2	p `domain` u						
	$v\ p\ w$		v `type` u			P!	
rdfs3	p `range` u						
	$v\ p\ w$	$w \in U \cup B$	w `type` u			P!	
rdfs4a	$v\ p\ w$		v `type Resource`	P	P	P	P
rdfs4b	$v\ p\ w$	$w \in U \cup B$	w `type Resource`	P	P	P	P
rdfs7x	p `subPropertyOf` q						
	$v\ p\ w$	$q \in U \cup B$	$v\ q\ w$			P_{NC}!	P

Table 2. Preserved P-Entailments (P = Rule preservation, P_{NC} = Non-contextual rule preservation, ! = Risk of undesirable inference)

Rule	Condition	Constraint	Conclusion	Reif.	N-Ary	S.P.	NdF
rdfp1	p `type` `FunctionalProperty`						
	$u\ p\ v$						
	$u\ p\ w$	$v \in U \cup B$	v `sameAs` w			P!	
rdfp2	p `type` `InverseFunctionalProperty`						
	$u\ p\ w$						
	$v\ p\ w$		v `sameAs` w			P!	
rdfp3	p `type` `SymmetricProperty`						
	$v\ p\ w$	$w \in U \cup B$	$w\ p\ v$			P_{NC}!	P
rdfp4	p `type` `TransitiveProperty`						
	$u\ p\ v$						
	$v\ p\ w$		$u\ p\ w$			P_{NC}!	P
rdfp5a	$v\ p\ w$		v `sameAs` v	P	P	P	P
rdfp5b	$v\ p\ w$	$w \in U \cup B$	w `sameAs` w	P	P	P	P
rdfp8ax	p `inverseOf` q						
	$v\ p\ w$	$w,q \in U \cup B$	$w\ q\ v$			P_{NC}!	P
rdfp8bx	p `inverseOf` q						
	$v\ q\ w$	$w \in U \cup B$	$w\ p\ v$			P_{NC}!	P
rdfp11	$u\ p\ v$						
	u `sameAs` u'						
	v `sameAs` v'	$u' \in U \cup B$	$u'\ p\ v'$	P	P	P_{NC}!	
rdfp14a	v `hasValue` w						
	v `onProperty` p						
	$u\ p\ w$		u `type` v			P!	
rdfp14bx	v `hasValue` w						
	v `onProperty` p						
	u `type` v	$p \in U \cup B$	$u\ p\ w$	P_{NC}	P_{NC}	P_{NC}	P_{NC}
rdfp15	v `someValuesFrom` w						
	v `onProperty` p						
	$u\ p\ x$						
	x `type` w		u `type` v			P!	
rdfp16	v `allValuesFrom` w						
	v `onProperty` p						
	u `type` v						
	$u\ p\ x$	$x \in U \cup B$	x `type` w			P!	

happens when the *RDF representation of annotated statements* function annotates at least one triple of the condition, and either we have non-contextual rule preservation, or we have rule preservation but the function does not annotate the triple in the conclusion. This is actually what happens with the Singleton Property for the rules `rdfs2`, `rdfs3`, and `rdfs7x` from the D*-entailments ruleset, and rules `rdfp1`, `rdfp2`, `rdfp3`, `rdfp4`, `rdfp8ax`, `rdfp8bx`, `rdfp11`, `rdfp14a`, `rdfp15`, and `rdfp16` (identified in the table with an exclamation mark), due to the RDFS interpretation that considers the singleton property as belonging to the extension of the original property [11, Sect. 3]. While there is no problem if the annotated fact is universally true (*i.e.*, we just want to provide additional information about a fact), it leads to undesirable conclusions when the context of the annotation is related with the identity of the resources (such as provenance or trust contexts), where we want to express that something is true only according to a source, or with a degree of confidence. For instance, let us suppose a functional property `birthplace` that we want to use in the context of provenance. It can be desirable to model that Barack Obama was born in the United

States according to a source, but in Kenya according to a different source. In this case the rule `rdfp1` would infer that the United States and Kenya are the same place in the non-annotated graph when using the Singleton Property.

It can be seen that Reification and N-Ary relations show poor preservation of rules, where most of those rules could be considered tautologies. The Singleton Property provides a mixture of rule preservation and non-contextual rule preservation for all the rules, that can be useful when we want to annotate universally true facts, but it is not usable when we want to have contextual information that is not universally true. NdFluents, by contrast, has neither non-contextual rule preservation nor rule preservation that can lead to undesirable inferences for any rule. There is only one rule where NdFluents is surpassed by the other approaches. Rule `rdfp11` presents *Rule Preservation* for Reification and N-Ary relations, but no rule preservation at all for NdFluents.

In addition, for the rules where NdFluents has no rule preservation, we observe that different conclusions hold, where we entail contextual knowledge. In Table 3 we see the conclusions for that set of rules with their conclusions. We can observe that the individual used in the annotation is entailed in the conclusion. For instance, let us suppose a property `capitalOf` with a domain of `PopulatedPlace`; if we state that Babylon was the capital of the Babylonian empire between 609 BC and 539 BC, instead of inferring that Babylon *is* a populated place (as a universal truth), we entail that Babylon between 609 BC and 539 BC was a populated place.

Table 3. Conclusions for rules with no rule preservation for NdFluents

Rule	Conclusion	Rule	Conclusion	Rule	Conclusion
rdfs2	v_c `type` u	**rdfp1**	v_c `sameAs` w_c	**rdfp14a**	u_c `type` v
rdfs3	w_c `type` u	**rdfp2**	v_c `sameAs` w_c	**rdfp15**	u_c `type` v

7 Related Work

In the original 4dFluents paper there were some issues not addressed by the authors. Later works have tried to identify and address those issues. Zamborlini and Guizzardi [17] present an alternative work to 4dFluents, where they present two different alternatives to represent temporally changing information in OWL. Both approaches have a similar model to Welty and Fikes's, where the entities are sliced for different times. The main difference is that in the first one, *Individual Concepts and Rigidity*, the original individuals are considered as classes. Thus, they are not described by any property, and a new slice has to be created every time that a property changes. On the other hand the second approach, "Objects and Moments", is based on *Relators* and *Qua-individuals* [9], where the individuals are represented by an entity, and their slices inherit its properties. Then, any time a property changes, it is reflected in the original entity. The first approach is more prone to the proliferation of timeslices, and can only

guarantee the immutability of original properties only by repetition on every timeslice. The second approach solves those issues at the cost of blurring the details of the changes of individual properties, and it is not clear how inheritance works in OWL. In a later work [18], Zamborlini and Guizzardi focus on solving the issues of the prior approaches for representing events and properties of individuals. They maintain the fluent-like representation for events, but move to an N-ary representation for properties. However, they still not address the possibility to have more that one domain relation, nor address how inheritance is performed in OWL.

There are also other works that compare the different approaches to represent contextual information. Gangemi and Presutti [2] present and compare a number of design patterns to represent N-Ary relations, including Reification and Context Slices [15], to represent additional information on binary relations. The comparison is done in four qualitative dimensions (DL reasoning support, polymorphism support, relation footprint, and intuitiveness) and five quantitative dimensions (number of needed axioms, expressivity, consistency checking time, classification time, and amount on newly generated constants). However, they only provide a brief outline of the reasoning power of each approach, while we are interested in more fine-grained comparison of entailment preservations. Scheuermann et al. [13], on the other side, perform a qualitative research that compares user preferences and ability for using different design patterns. In their study the fluents pattern is regarded as the most complicated and less used to model, while making a temporal slice of the predicate (which could be represented using the Singleton Property in RDF) seems more intuitive. The N-ary pattern is the model most frequently used. The model regarded as the most user-friendly is not representable using OWL, because it requires having a predicate as an argument of another (an approximation in RDF could be using N-Quads, though). Hernández et al. [5] compare Reification, N-Ary relations, Singleton Properties and Named Graphs to encode Wikidata in practice. They provide space requirements and query performance for each approach in 4store[3], BlazeGraph[4], GraphDB[5], Jena TDB[6] and Virtuoso[7]. They report that Singleton Properties provide the most concise representation on a triple level, while N-Ary predicates is the only model with built-in support for SPARQL property paths. In addition, the Singleton Property usually lacks performance due to the number of predicates, whereas there is no clear winner among the other approaches. Virtuoso exhibits the best performance, while Jena and 4store show the worst results. Later, Hernández et al. [6] extend their previous work to compare Virtuoso, BlazeGraph, Neo4J[8], PostgreSQL[9] with a set of new experiments, based on the idea of performing sets of lookups for atomic patterns with exhaustive combinations of constants and

[3] https://github.com/garlik/4store.
[4] https://www.blazegraph.com.
[5] http://graphdb.ontotext.com.
[6] https://jena.apache.org/documentation/tdb.
[7] https://virtuoso.openlinksw.com.
[8] https://neo4j.com.
[9] https://www.postgresql.org.

variables, in order to give an idea of the low-level performance of each configuration. In this set of experiments standard reification and named graphs performed best, with N-Ary relations following in third, and singleton properties not being well-supported.

8 Conclusions

Representing annotations on multiple dimensions is a current challenge in RDF and OWL. We have proposed the NdFluents ontology, a multi-dimension annotation ontology, based on 4dFluents. To the best of our knowledge, this is the first generic extension of 4dFluents for an arbitrary combinations of context dimensions. This representation is intended to be extended in a modular way for each desired dimension. In addition, we have presented three design patterns and additional considerations to keep in mind when modeling data with NdFluents. We study how many of the original inference rules are preserved when annotating the data with NdFluents and compare with the main approaches to annotate data: Reification, N-Ary Relations, and Singleton Property. The results show that NdFluents preserves more desirable entailments, while omitting undesirable entailments, than any alternative. The Singleton property presents non-contextual rule preservation for many of the rules, and can lead to undesirable entailments when the annotated facts are not universally true. Reification and N-Ary relations preserve the fewest number of entailment rules.

Lines of future work are manifold: First, we want to apply this model to real world datasets. Our goal is to exploit the context of information to make the datasets fit for question answering, as well as determine the most relevant data sources. This includes providing additional information based on the context and helping to find the most trustworthy data for the answer. Second, we intend to look deeper into the entailment preservations for different approaches using bigger subsets of OWL 2, such as OWL LD and OWL 2 RL/RDF, and possible reformulations of the approaches that could improve the results. Third, we plan to perform an experimental evaluation of the different annotation models using different triple stores wrt different factors, such as size, loading time, query response time, and query formulation complexity.

Acknowledgements. This work is supported by funding from the EU H2020 research and innovation program under the Marie Skłodowska-Curie grant No 642795, and from ANR grant 14-CE24-0029 for project OpenSensingCity. Authors would like to thank Chris Welty for his supportive comments, and Amro Najjar for his suggestions.

References

1. Carothers, G.: RDF 1.1 N-Quads: a line-based syntax for RDF datasets. W3C Recommendation (2014). https://www.w3.org/TR/n-quads
2. Gangemi, A., Presutti, V.: A multi-dimensional comparison of ontology design patterns for representing n-ary relations. In: Emde Boas, P., Groen, F.C.A., Italiano, G.F., Nawrocki, J., Sack, H. (eds.) SOFSEM 2013. LNCS, vol. 7741, pp. 86–105. Springer, Heidelberg (2013). doi:10.1007/978-3-642-35843-2_8

3. Giménez-García, J.M., Zimmermann, A., Maret, P.: NdFluents: A multi-dimensional contexts ontology. Technical report, Université Jean Monnet (2016)
4. Hartig, O., Thompson, B.: Foundations of an alternative approach to reification in RDF. CoRR (2014). http://arxiv.org/abs/1406.3399x
5. Hernández, D., Hogan, A., Krötzsch, M., Reifying, R.D.F.: What works well with wikidata? In: 14th International Semantic Web Conference (ISWC) (2015)
6. Hernández, D., Hogan, A., Riveros, C., Rojas, C., Zerega, E.: Querying wikidata: comparing SPARQL, relational and graph databases. In: Groth, P., et al. (eds.) ISWC 2016. LNCS, vol. 9982, pp. 88–103. Springer, Cham (2016). doi:10.1007/978-3-319-46547-0_10
7. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M., et al.: SWRL: A semantic web rule language combining OWL and RuleML. W3C Member Submission (2004). http://www.w3.org/TR/2004/REC-owl-ref-20040210/
8. Kifer, M., Boley, H.: RIF overview. W3C Working Draft, W3C, October 2009 (2013). http://www.w3.org/TR/rif-overview
9. Masolo, C., Guizzardi, G., Vieu, L., Bottazzi, E., Ferrario, R.: Relational roles and qua-individuals. In: AAAI Fall Symposium on Roles, an Interdisciplinary Perspective (2005)
10. Nardi, D., Brachman, R.J., et al.: An introduction to description logics. In: The Description Logic Handbook: Theory, Implementation, and Applications (2003)
11. Nguyen, V., Bodenreider, O., Sheth, A.P.: Don't like RDF reification? Making statements about statements using singleton property. In: 23rd International Conference on World Wide Web (2014)
12. Noy, N., Rector, A., Hayes, P., Welty, C.: Defining n-ary relations on the semantic web. W3C Working Group Note (4) (2006). https://www.w3.org/TR/swbp-n-aryRelations/
13. Scheuermann, A., Motta, E., Mulholland, P., Gangemi, A., Presutti, V.: An empirical perspective on representing time. In: 7th International Conference on Knowledge Capture (2013)
14. ter Horst, H.J.: Completeness, decidability and complexity of entailment for RDF schema and a semantic extension involving the OWL vocabulary. J. Web Seman. **3**(2–3), 79–115 (2005)
15. Welty, C., Slices, C.: Representing contexts in OWL. In: Workshop on Ontology Patterns (2010)
16. Welty, C., Fikes, R.: A reusable ontology for fluents in OWL. In: 1st International Conference of Formal Ontology in Information Systems (FOIS) (2006)
17. Zamborlini, V., Guizzardi, G.: On the representation of temporally changing information in OWL. In: Enterprise Distributed Object Computing Conference Workshops (EDOCW) (2010)
18. Zamborlini, V., Guizzardi, G.: An ontologically-founded reification approach for representing temporally changing information in OWL. In: 11th International Symposium on Logical Formalizations of Commonsense Reasoning (COMMONSENSE) (2013)
19. Zimmermann, A.: RDF 1.1: On semantics of RDF datasets (2014). https://www.w3.org/TR/rdf11-datasets/