July 2016

NdFluents: A Multi-dimensional Contexts Ontology

José M. GIMÉNEZ-GARCÍA [a], Antoine ZIMMERMANN [b] and Pierre MARET [a]
[a] *Université de Lyon, CNRS, UMR 5516, Laboratoire Hubert-Curien, Saint-Étienne, France*
[b] *École Nationale Supérieure des Mines, FAYOL-ENSMSE, Laboratoire Hubert Curien, F-42023 Saint-Étienne, France*

Abstract. Annotating semantic data with meta-data is becoming more and more important to provide information about the statements being asserted. While initial solutions proposed a data model to represent a specific dimension of meta-information (such as time or provenance), the need for a general annotation framework which allows representing different context dimensions is needed. In this paper, we extend the 4dFluents ontology by Welty and Fikes—on associating temporal validity to statements—to any dimension of context, and discuss possible issues that multi-dimensional context representations have to face and how we address them.

Keywords. Ontologies, OWL, Context, Reification

1. Introduction

In knowledge representation, it is often necessary to characterize the context associated to a statement, such as when and how it was generated, or who uttered it. Many models exist for providing statements about statements, some of which are specific to a certain "dimension" of context, such as temporal validity.

Along these lines, in 2006, Welty and Fikes [1] proposed an ontology for describing fluents (*i.e.*, entities whose characteristics change over time). Their approach advanced the state of the art in temporal representation, and has been used and extended in other works, but it only addresses one dimension characterizing a statement. Nonetheless, Welty and Fikes's approach can be reused to provide a solution for arbitrary dimensions of context. We generalize it in a generic ontology that can then be extended to implement a specific dimension of context. In addition, we address the problem of modeling contextual datatype properties, as well as combining different contextual dimensions. This work is motivated by the need to characterize web datasets in terms of trust, provenance, and temporal validity, in the context of a question answering system within project WDAqua.[1]

In reference to the original 4dFluents ontology, we call our ontology NdFluents. We recall Welty and Fikes's contribution in Section 2, then we present the NdFluents

[1] `http://wdaqua.informatik.uni-bonn.de`

ontology in Section 3 together with some issues. In Section 4, we summarize guidelines for expressing contextualized statements and apply this to a concrete use case that we published according to the Web of Data best practices. We then compare our approach to related work on representing the context of information in Section 5 before concluding.

2. 4dFluents Ontology

Welty and Fikes [1] 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 different 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 [1])

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

Welty and Fikes argue that, although there are various other ways of modelling fluents, the 4dFluents approach has proved being efficient in projects led by the authors. Based on these observations, we in turn think that adopting a similar approach, generalized to any dimension of context, would be a sensible choice.

3. Extension of 4dFluents Ontology for Multiple Dimensions

In this section we discuss how to broaden the 4dFluents ontology to other dimensions different than time. In the first subsection we propose our ontology, then we proceed to discuss different representation details in the following subsections.

3.1. Extending the 4dFluents 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 *"A 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 to represent complex information, such as: *"A temporal part 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 any context dimension in the *NdFluents* ontology. The ontology, shown in Ontology 3, and published in `http://www.emse.fr/~zimmermann/ndfluents.html`, is a direct extension from temporal parts to contextual parts.

Note that `FunctionalObjectProperty( nd:contextualExtent )` axiom is not included. 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 in the general case when we have more than one context dimension. Depending on the model used to represent the information, it will be necessary to add it (see Section 3.3).

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

```
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: NdFluents ontology

concrete implementation of NdFluents, as we show in Ontology 4. In Figure 1 we show the representation of a statement with temporal context using this ontology. The non-dashed parts are equivalent to the original 4dFluents ontology, while the dashed parts correspond to the NdFluents extension. Other context 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 counterpart in the provenance dimension.

3.2. Dealing with Datatype Properties

The original 4dFluents ontology does not provide any information for modelling datatype properties. While there is nothing that prevents using regular datatype properties with contextual parts of an entity, it may be desirable to declare explicit axioms for context properties to facilitate reasoning on that information. In that case, the statements of Ontology 5 need to be added to the NdFluents ontology. Figure 2 shows an example where a contextual property is used to state the population of Paris in a specific temporal interval. Note that it is possible to create specific contextualProperty subproperties for different contextual dimensions (*i.e.*, `temporalProperty` for `TemporalPart`) for properties related to concrete context dimensions.

3.3. Combining Different Context Dimensions

An important scenario where NdFluents becomes relevant is when the necessity of combining two or more dimensions of context arises, such as saying that *"according to Wikipedia, Paris is the capital of France since 508"*. The NdFluents ontology supports different ways of representing the information. In this section we describe the most relevant models that can be used.

```
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( :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

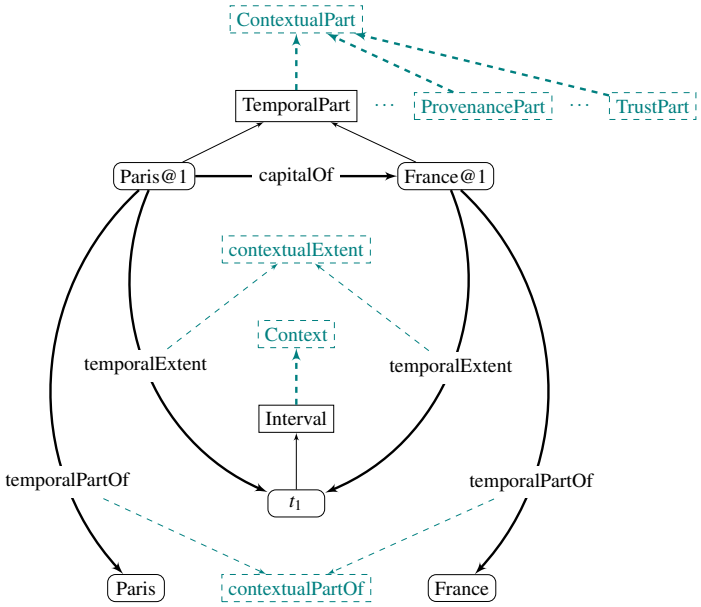

Figure 1. Example of 4dFluents as implementation of NdFluents

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

Ontology 5: Datatype axioms for NdFluents ontology

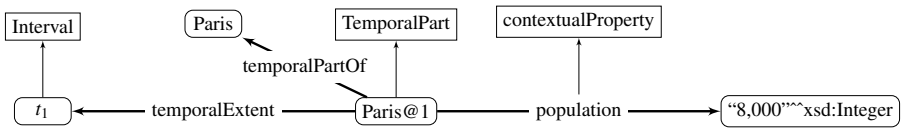

Figure 2. Example of Contextual Datatype Property

Contexts in Context. One possible model to represent information using different context dimensions is to relate a Contextual Part to another Contextual Part. This approach can be taken when the "first level" Contextual Parts are relevant facts of the knowledge base, and the intention is to state additional information about them. To be able to reason about different contextual levels of any entity, it is desirable for the `contextualPartOf` property to be transitive, which can be achieved by adding the axiom of Ontology 6.

While data about different context dimensions can be more fine-grained using this model, it also grows in complexity. For example, in Figure 3 the statement `capitalOf` is related to the provenance dimension Contextual Part `Paris@1.1`. This information is in no way related to the Temporal Part `Paris@1`. While we could have this statement duplicated in the example, this is not possible as soon as we add another provenance Contextual Part to `Paris@1.1`. We believe that this model 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 6: Transitive axiom for NdFluents ontology

Use Multiple Contexts for each Contextual Part. A more generic approach for representing entities with more than one contextual dimension is to have Contextual Parts with more than one Contextual Extent. Using this model, only one Contextual Part is created for a combination of context dimensions. This Contextual Part is then related to all the related contextual information, as shown in Figure 4. This model is easier to model: Relating the Contextual Part with the context dimensions is straightforward. It also avoids ambiguity when modelling contextual information related to more than one contextual dimension, and reduces the number of resources in the ontology (*i.e.*, while the previous model needed one Contextual Part for each context dimension involved, this approach only requires one Contextual Part). Note that `contextualPartOf` is a functional property, which means that there cannot be a Contextual Part of more than one entity.

Combine Different Contexts on one Contextual Extent. Finally, a third possibility is to create Contextual Extents that combine two or more context dimensions, and enforce a limit of only one Contextual Extent per Contextual Part. This model adds a layer of complexity to the previous approach, but it can be useful to require a specific combination of context dimensions on a set of Contextual Parts. This can be achieved by adding the axiom in Ontology 7. We show an example of this approach on Figure 5. 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.*,

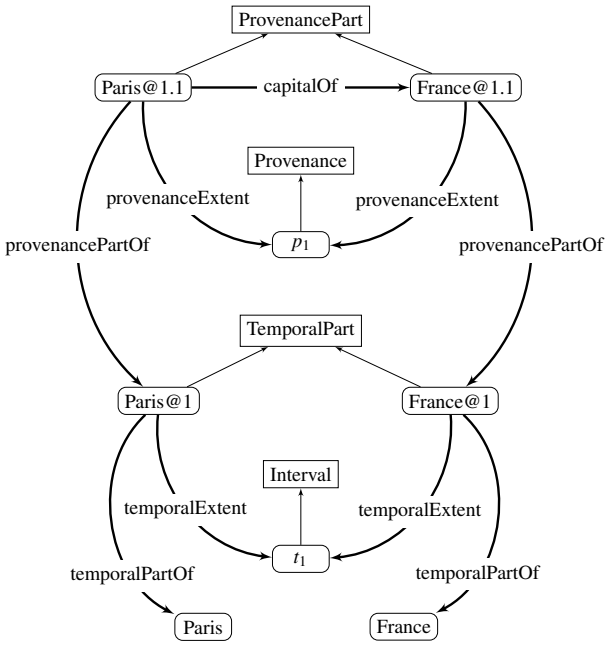

Figure 3. Contexts in Context

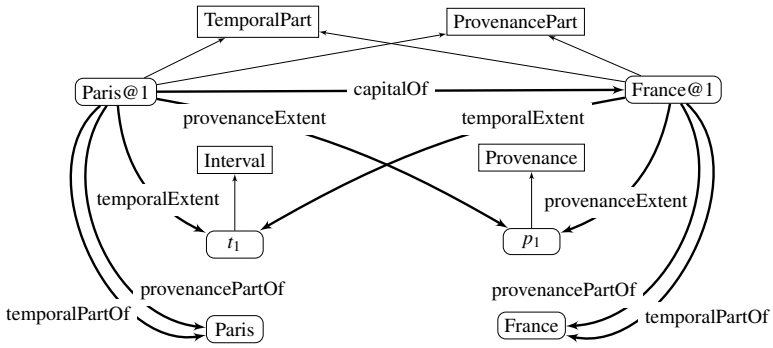

Figure 4. Multiple contexts on one ContextualPart

`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.

3.4. Relations between Different ContextualParts

The NdFluents ontology presented thus far allows to model relations among different contextual parts of different context dimensions (*i.e.*, a Temporal Part of Paris could be the capital of a Provenance Part of France). While this can be convenient for individual cases, it is often needed for a contextual property to be related to Contextual Parts of the same dimension of context. In this case, it is necessary to add the appropriate axioms

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

Ontology 7: Functional Contextual Extents axiom for NdFluents ontology

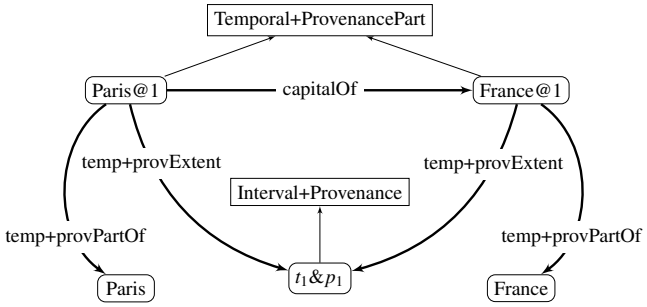

Figure 5. Combination of different contexts on one contextual extent

to the ontology. In Ontology 8 we show the needed axioms to include this restriction on the Temporal Parts. 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 )
)
```

Ontology 9: Temporal restriction on datatype properties 4dFluents ontology

In a similar fashion, it is usually desirable that Contextual Parts of the same contextual dimension relate to the same Contextual Extent. That is, if a Provenance Part of Paris relates to a Provenance Part of France, their `provenanceExtent` properties should have the same Provenance object. However, this restriction cannot be expressed in OWL.

If needed, a rule language (such as SWRL [2] or RIF[2]) can be used for this purpose, but this case goes beyond the scope of this paper.

3.5. Adapting Non-Contextual Ontologies to NdFluents

In the previous sections, we show how to define and use NdFluents to define properties that apply to contextual parts. However, in some cases, we would rather reuse existing ontologies. It is not always possible to add a subproperty relationship between an already defined property and a contextual property because most restrictions will not hold for the contextual parts of an entity. For instance, let us suppose that the property `capitalOf` has domain `City`, and we use it as a fluent property in Figure 1. Then, it would be inferred that `Paris@1` is a city, instead of `Paris` being inferred as a city.

To address this situation, instead of using the original properties, it is necessary to create new properties that are somehow related to the original. It is then necessary to model the new properties using Class Expressions for the restrictions. We show an example for the domain and range of the property `capitalOf` in Ontology 10.

```
1 Declaration( ObjectProperty( ex:contextualCapitalOf ) )
2 SubObjectPropertyOf( ex:contextualCapitalOf 4d:fluentProperty )
3 ObjectPropertyDomain( nd:contextualCapitalOf ObjectAllValuesFrom( ex:
      contextualProperty ex:City) )
4 ObjectPropertyRange( nd:contextualCapitalOf ObjectAllValuesFrom( ex:
      contextualProperty ex:Country ))
```

Ontology 10: Definition of Related Contextual Property to `capitalOf`

3.6. Dealing with Terminological Statements

In general, contexts are used just for assertions (what is usually considered the *ABox*). However, there are cases where terminological statatements (the *TBox*) can also be viewed under different context dimensions. Consider the case of classic biological kingdoms. While in the U.S. it has been traditionally considered that there are six kingdoms (*Animalia, Plantae, Fungi, Protista, Archaea/Archaeabacteria, and Bacteria/Eubacteria*), many other countries consider only five (*Animalia, Plantae, Fungi, Protista and Monera*). This classification evolved from the historic view of animal and plant kingdoms, while more recent classifications include up to eight kingdoms. Whether to include viruses in the taxonomy is still an ongoing debate.

Such classification would induce `subClassOf` relations such as `Haloarchaea subClassOf Archaeabacteria` and `Haloarchaea subClassOf Monera`, which hold for different temporal and provenance domains. However, to make these statements contextual, it is necessary to define a contextual subproperty related to `subClassOf`. This is possible, but it is important to take into account that the created property will not benefit from the standard inferences associated with subClassOf. There is no solution using only OWL that allows us to perform that kind of inference.

[2] `https://www.w3.org/TR/rif-overview`

4. NdFluents in Practice

In this section, we concretize the previous information on actual steps to implement the NdFluents ontology in practice, with a focus on the decisions to make in each step, and our recommendation. Then, we proceed to demonstrate an actual implementation of the NdFluents ontology for a concrete set of data with different context dimensions following those steps.

4.1. Modeling a Knowledge Base

In order to model a knowledge base with a number of context dimensions, it is necessary to model the ontology in the TBox, and then create the statements in the ABox using the ontology. The ontology can be modeled according to the following steps:

1. For each context dimension, create the appropriate subclass of `ContextualPart`, and a subclass of `Context` (such as `TemporalPart` and `Interval` for temporal dimension).
2. For each context dimension, create a subproperty of `contextualExtent` and `contextualPartOf` (such as `provenanceExtent` and `provenancePartOf` for Provenance dimension).
3. If any contextual part includes datatype properties, it has to be decided whether to use the datatype restriction axioms (Ontology 5). We advise to include them to improve reasoning capabilities.
4. If there is more than one context dimension in the ontology, the model to represent the information needs to be selected (see Section 3.3). We recommend to use the second approach (*use different Contexts for each Contextual Part*, see Figure 4). If *Contexts in Context* (Figure 3) is used, the transitivity axiom to the `contextualPartOf` property (Ontology 6) needs to be added. If *combine different contexts in Contextual Extents* (third approach) is used, create the combined subclasses of `ContextualPart` and `Context`, and the combined subproperties of `contextualExtent` and `contextualPartOf`, as shown in Figure 5.
5. If there is more than one context dimension, it is possible to use the same contextual properties for every dimension or create a different subproperty of `contextualProperty` for each one (see Section 3.4). For the last case, the context restrictions axioms (see Ontology 8 for temporal dimension) need to be added for each dimension. We recommend to include these axioms.

Once the ontology is modeled, the next series of steps are needed to model the contextual statements:

1. For each context dimension, it is necessary to create the related Context information. This is a new resource of the related `Context` subclass and the adequate information (interval for temporal context, provenance information for provenance context, *etc.*). Depending on the model you choose to represent the information, it is needed to create a different resource for each context of an entity (Figures 3 and 4), or a unique resource that combines the contexts (Figure 5).
2. For each entity, create its related Contextual Parts. These are new resources of type the appropriate `ContextualPart` subclasses (*e.g.*, `TemporalPart`, `ProvenancePart`). This resources are connected to their non-contextual entity

by `contextualPartOf` subproperties (`temporalPartOf`, `provenancePartOf`), and to the information related to the context, modeled as Context resources (*i.e.*, `Interval`, `Provenance`), by `contextualExtent` subproperties (`temporalExtent`, `provenanceExtent`.).

In the following section, we present an example of a concrete use case where we follow those steps to model a knowledge base.

4.2. Practical Use Case

In this section the NdFluents ontology is used on a practical example with two contextual dimensions: The estimated evolution of Earth population according to different sources, which needs temporal and provenance dimensions. For this task, we use the information provided by Wikipedia[3].

We model the TBox according to the steps defined in the previous section for the Temporal and Provenance dimensions. As the population of each period will be defined as datatype properties, we decide to include the datatype restriction axioms (Ontology 5 for temporal dimension, and similarly for the provenance dimension) in step 3. In step 4, we choose to *use multiple Contexts for each Contextual Part* (Figure 4) to model the information. The resulting TBox will be comprised of ontologies 3, 4, 5, and the corresponding implementation of NdFluents for provenance (equivalent to Ontology 4).

The contextual statements are also modeled following the steps of previous section. In step 1, we create the Intervals and the Provenance extents for every period and source. To model Intervals we use the OWL-Time ontology[4], while for Provenance Extents we use the PROV-O ontology[5] (along with OWL-Time and the Event ontology[6] to model the Activity). In step 2, we create the contextual parts corresponding to each period and source for the Earth. Those parts are of type `TemporalPart` and `ProvenancePart`, are defined as `temporalPartOf` and `provenancePartOf` of `Earth`, and are connected to their corresponding Intervals and Provenance Extents by `temporalExtent` and `provenanceExtent` properties. Finally, they include the population as a datatype property (note that in Wikipedia a few number of population values are given as an interval, in which case we use the average value).

The complete dataset is published in `http://www.emse.fr/~zimmermann/ndfluents.html`

With this data is possible to make queries using SPARQL about specific periods and sources, obtaining individual or aggregated data. For example, it is possible to obtain the average population and number of studies per year using the Query 1. Using this result one could, for every period with at least two studies, compute the p-value Student's t-distribution considering the population of each study as theoretical mean. This value then could be attached to the contextual part as a new Trust Extent.

[3] `https://en.wikipedia.org/wiki/World_population_estimates`
[4] `https://www.w3.org/TR/owl-time`
[5] `https://www.w3.org/TR/prov-o`
[6] `http://motools.sourceforge.net/event/event.html`

```
SELECT ?year (AVG(?population) AS ?average) (COUNT(?earth_part) AS ?count)
WHERE {
       ?earth_part nd:temporalExtent [
           time:intervalDuring [
               time:hasDateTimeDescription [ time:year ?year ]]] ;
           dbo:populationTotal ?population .
}
GROUP BY ?year
```

Query 1: Extracting statistical data for year zero

5. Related work

NdFluents is not the first extension of 4dFluents. There are a number of extension for Spatio-Temporal representations. Batsakis and Petrakis [3, 4] enhance the 4dFluents mechanism with qualitative temporal expressions to represent relations between intervals, and allow the definition of new intervals using this relations. Later, they extend it with several types of qualitative spatial relations and use it for their SOWL query language. Milea et al. develop *tOWL* [5], an extension of OWL for temporal domains, on top of a 4dFluents layer. Harbelot et al. [6, 7] use tOWL and GeoSPARQL for spatio-temporal representations of entities. However, to the best of our knowledge, NdFluents *is* the first generic extension of 4dFluents for any number of arbitrary context dimensions.

There are other approaches to model arbitrary contextual information about entities or statements. Reification[7] is the standard W3C model to represent information about an statement, proposed in 2004. A statement is represented as an instance of `rdf:statements`, which relates to the original triple with the properties `rdf:subject`, `rdf:predicate` and `rdf:object`. This model can be seen in Figure 6. However, reification lacks formal semantics to connect the original triple with the reified statement, which disallows any reasoning on the information.

N-Ary relations[8] 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. Figure 7 shows an example of an N-ary relation. While N-ary relation are an improvement over reification, it does not allow complete OWL inference. For instance, it is not possible to perform any reasoning involving inverse, transitive or symmetric relationships. Wikidata[9] makes use of an specific implementation of N-ary relations, where each entity is related with statement, that in turn can be related to values, qualifiers, or references. The estimated word population, with temporal qualifiers for the date, and references for the provenance, is modeled using this pattern in Wikidata[10].

The Singleton Property [8, 9] is a recent proposal to represent information about statements in RDF. An particular instance of the predicate is created for every triple. This instance is related to the original predicate by the `singletonPropertyOf` property. Then, each statements can be unequivocally referenced using its predicate for attaching additional information. An example of this model can be seen in Figure 8. The Singleton

[7] `http://www.w3.org/TR/2014/REC-rdf11-mt-20140225/#reification`
[8] `https://www.w3.org/TR/swbp-n-aryRelations`
[9] `https://www.wikidata.org`
[10] `https://www.wikidata.org/wiki/Q2#P1082`

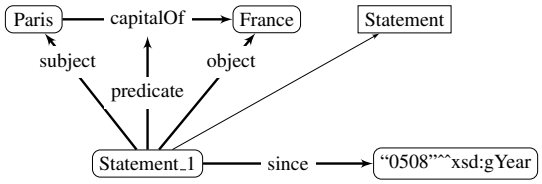

Figure 6. Representation of a ternary statement using reification

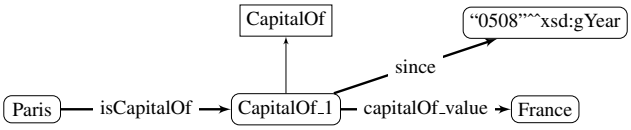

Figure 7. Representation of a ternary statement using N-ary relations

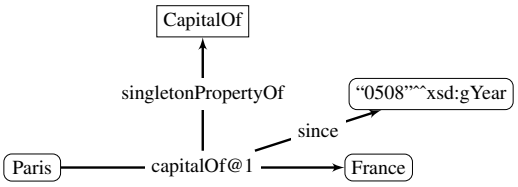

Figure 8. Representation of a ternary statement using Singleton Property

Property is an intuitive approach, but it cannot be expressed in OWL and needs to extend the RDF semantics to make reasoning possible.

6. Conclusion

Representing contextual information in different dimensions is a current challenge in OWL. We have proposed NdFluents, a multi-domain contextual representation, based on the 4dFluents ontology. This is to the best of our knowledge, the first generic extension of 4dFluents for any number of arbitrary contextual dimensions. This representation is intended to be extended in a modular way for each desired context dimension in the ontology. We have discussed different possible models that can be used when combining different dimensions, possible additions to consider depending on the data we want to represent, and open issues of the ontology. We have also provided schematic guidelines for using NdFluents and a practical example. Both the ontology and the example are published for public usage.

As future work, 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 a 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. A particular challenge lies in finding the temporal validity of the facts found in the data.

7. Acknowledgement

This project is supported by funding received from the European Unions Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 642795.

References

[1] Christopher A. Welty and Richard Fikes. A Reusable Ontology for Fluents in OWL. In *Proc. of FOIS 2006*, pages 226–236.

[2] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A Semantic Web Rule Language - Combining OWL and RuleML. Technical report, 2004. URL `http://www.w3.org/TR/2004/REC-owl-ref-20040210/`.

[3] Sotiris Batsakis and Euripides GM Petrakis. Representing temporal knowledge in the semantic web: The extended 4d fluents approach. In *Combinations of Intelligent Methods and Applications*, pages 55–69. Springer, 2011.

[4] Sotiris Batsakis, Kostas Stravoskoufos, and Euripides G. M. Petrakis. Temporal reasoning for supporting temporal queries in OWL 2.0. In *Proc. of KES 2011*, pages 558–567, 2011.

[5] Viorel Milea, Flavius Frasincar, and Uzay Kaymak. tOWL: A temporal web ontology language. *IEEE Trans. on Sys., Man, and Cybernetics, Part B*, 42(1):268–281, 2012.

[6] Benjamin Harbelot, Helbert Arenas, and Christophe Cruz. Continuum: a spatiotemporal data model to represent and qualify filiation relationships. In *Proc. of IWGS 2013*, pages 76–85, 2013.

[7] Benjamin Harbelot, Helbert Arenas, and Christophe Cruz. The spatio-temporal semantics from a perdurantism perspective. In *Proc. of GEOProcessing 2013*, 2013.

[8] Vinh Nguyen, Olivier Bodenreider, and Amit P. Sheth. Don't like RDF reification?: making statements about statements using singleton property. In *Proc. of WWW 2014*, pages 759–770, 2014.

[9] Vinh Nguyen, Olivier Bodenreider, Krishnaprasad Thirunarayan, Gang Fu, Evan Bolton, Núria Queralt-Rosinach, Laura Inés Furlong, Michel Dumontier, and Amit P. Sheth. On reasoning with RDF statements about statements using singleton property triples. *CoRR*, abs/1509.04513, 2015. URL `http://arxiv.org/abs/1509.04513`.