Representation of tensed relations in OWL
A survey of design patterns *

Paweł Garbacz and Robert Trypuz

John Paul II Catholic University of Lublin
Department of the Foundations of Computer Science
Faculty of Philosophy
Al. Racławickie 14, Lublin, Poland
{garbacz,trypuz}@kul.pl

Abstract. The topic of this paper are the so-called tensed relations, i.e., those relations that hold between objects with respect to time. As tensed relations are not, almost by definition, binary relations, they need a special treatment in the case of such formal languages as OWL where only binary relations are explicitly expressible. We study in this paper a number of ways in which this expressivity constraint can be worked around. The solutions we found range from simple, well-known reification patterns to philosophically sophisticated conceptions. Besides fleshing out the details of those patterns we compare them to one another to show their strengths and limitations in various usage scenarios.

1 Introduction

Except for mathematical knowledge and the like, most of the facts we know involve some kind of temporal references. Take a simple example. Anna was married to Paul from 2010 to 2014, but they divorced in January 2014. If we are to represent such facts in a formal language and if we have at our disposal the full power of, say, first-order logic, then our task is trivial. For example, using Common Logic's CLIF language we could grasp this change in the martial status of Anna by: "*and (married Anna Paul 2014) not(married Anna Paul 2015))*". The task becomes more demanding if the symbolic formalism of our choice is more restricted, e.g., when we are restricted to a DL language, say a language from the OWL family, where we do not have access to ternary and higher-arity relations.

This paper focuses on one specific sub-case of that problem, namely how to express tensed binary relations in OWL. By *tensed binary relation* we mean any ternary relation of the form $R(x, y, t)$, where one of its arguments, say the third one, refers to times (i.e., moments or periods). More generally speaking, we are interested in OWL representations of *tensed relation*, i.e. $(n + 1)$-ary relations $(n > 2)$ of the form $R(x_1, x_2, \ldots, x_n, t)$. [1]

* The research presented in this paper was supported by the *Ontological Foundations for Building Historical Geoinformation Systems* (2bH 15 0216 83) grant funded by National Programme for the Development of Humanities (http://nprh.org/).

[1] We borrow the term and the concept of tensed relation from [3].

The solutions we found can be classified into three main categories:

1. standard pattern(s), which are straightforward applications of the W3C recommendations;
2. simple pattern, which is an in-house pattern that does not involve any significant ontological commitments;
3. philosophically-laden patterns, which are patterns that require that you should accept some, more or less controversial, philosophical position.

Needless to say, the sample of solutions discussed below does not exhaust all conceptual possibilities, but we tried to collect all existing patterns and add a couple of new insights, hoping that such study may inspire further, more elaborated research. The solutions discussed here are partially determined by the context of this study, which is a research project on the historical geoinformation systems. One of the main conceptual issues we came across within this project is the problem of variability of human settlements (e.g., hamlets, villages, towns, etc.) and their identity through time. Human settlements change their types, names, even locations in time, so use of tensed relations is unavoidable.

To make the issue more palatable we will use in this paper the following running example. The contemporary city of Nieszawa was first mentioned in historical records around 1425 as located at N52°59.48′ E18°30.28′ – we will call this location "location 1". Around 1460 Nieszawa was relocated to N52°50.12′ E18°54.05′ – we will call this location "location 2". In short:

1. Nieszawa is located at location 1 from 1425 to 1459.
2. Nieszawa is located at location 2 from 1460 to the present day.

In each solution Nieszawa is represented as the OWL individual `:settlement1` and the two locations as `:location1` and `:location2`. Using this minimal dataset each pattern was implemented as an OWL ontology for the running example – all ontologies can be accessed from `http://onto.kul.pl/change/`.

Our work can be seen as a continuation of Welty and Fikes' research presented in [11]. In comparison to the Welty and Fikes' work we take into account a broader spectrum of possible patterns. While Welty and Fikes recommend, with certain caveats, the perdurantist approach, which we discuss in section 4.2 below, our approach seems to us more descriptive in nature.

On the other hand let us observe that such approaches as Time Ontology [1] answer different, although related questions. Namely, we are not focused here so much on the question "How to represent time (or such time objects as moments or intervals) in OWL?", but rather on the question "How to represent tensed relations in OWL?".

The paper is organised as follows. On the basis of the above classification, sections from 2 to 4 define the patterns for representation tensed relations in OWL. Section 5 compares these patterns to one another, indicating their strengths and limitations. Section 6 points towards future extensions of this research.

2 Standard Patterns

W3C document [9] presents a couple of strategies to represent n-ary relations ($n > 2$) in RDF and OWL. The patterns presented there can be adapted to represent change in the way we will show below.

The basic idea of the approach presented in [9] is that each n-ary relation between individuals is represented as a separate class. (Ontologically the classes created in this way are often called "reified relations".) So an instance of the relation—an n-tuple—is represented as an instance of the class plus additional n binary relations providing links to each argument of the n-tuple.

Thus if you need to represent that Nieszawa was located at N52°59.48′ E18°30.28′ from 1425 to 1459 within the OWL framework, you need (i) a *class* corresponding to location relation, let it be `:LocationRelation`, and (ii) then an instance of it, say `:settlement1_location_relation1`, and (iii) finally four binary relations to connect `:settlement1_location_relation1` with the exemplary arguments of location relation, i.e. with: `:settlement1`, `:location1`, 1425, and 1459. And if you need to represent the fact that Nieszawa has been located at N52°50.12′ E18°54.05′ from 1460, then you can reuse `:LocationRelation`, its new `:settlement1_location_relation2` instance, and three binary relations to link the latter to `:settlement1`, `:location2` and 1460.

2.1 Reification With No Distinguished Participant Pattern

In this pattern no participant of the reified relation is distinguished. i.e. as stated in [9], there is no "single individual standing out as the subject or the "owner" of the relation". In this approach all the links go from an instance of the reified relation to all the participants of the relation. In our example an instance `:settlement1_location_relation1` of `:LocationRelation` will be linked with the participants as shown below:[2]

```
:settlement1_location_relation1 a :LocationRelation ;
    :hasLocation :location1 ;
    :hasLocationParticipant :settlement1 ;
    :hasBeginning "1425-01-01T00:00:00"^^xsd:dateTime ;
    :hasEnd "1459-12-31T23:59:59"^^xsd:dateTime .
```

2.2 Reification With Distinguished Object Pattern

The previous pattern may not appear so compelling when one believes that `:settlement1` is not on a par with other participants but is a *distinguished* participant as the subject of the location relation. Then we could provide the following pattern:

```
:settlement1 a :MaterialObject ;
    :hasLocationRelation :settlement1_location_relation1 ,
    :settlement1_location_relation2 .
```

[2] For each pattern we shall provide a graph corresponding to the running example. The graph will be partially serialized in Turtle syntax and illustrated by a diagram compliant with the Graffoo standard (see `http://www.essepuntato.it/graffoo/`).

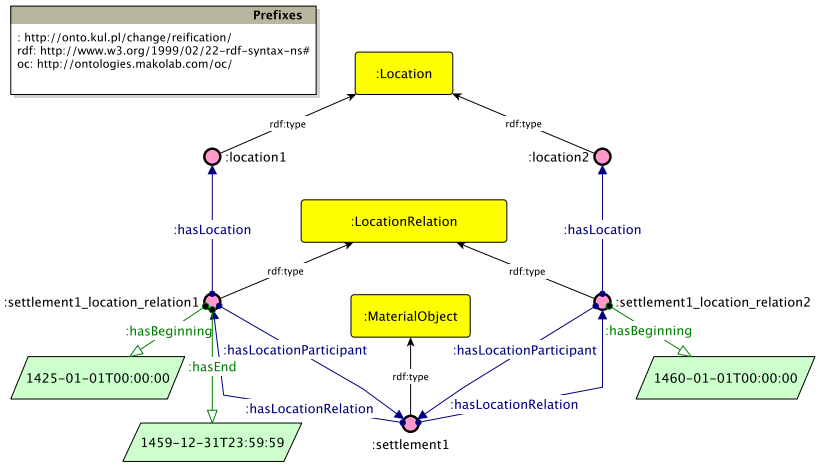


Fig. 1: Reification Pattern

```
:settlement1_location_relation1 a :LocationRelation ;
    :hasLocation :location1 ;
    :hasBeginning "1425-01-01T00:00:00"^^xsd:dateTime ;
    :hasEnd "1459-12-31T23:59:59"^^xsd:dateTime .
```

As the reader has already probably realised, the two reification patterns presented in sections 2.1 and 2.2 are isomorphic providing `:hasLocationRelation` is assumed to be the inverse of `:hasLocationParticipant`. For this reason we show both patterns in the same figure 1.

3 Simple Pattern

The simple pattern incorporates temporal references into OWL object properties so that there exists a separate object property for each time (moment or period) under consideration. In our case the pattern contains one OWL object property `:hasLocation` with two sub-properties: one for each period at question:

1. `:hasLocationAt1425-1459`, which is to contain all location relations that start in 1425 and end in 1459;
2. `:hasLocationAt1460-`, which is to contain all location relations that start in 1460.

Each of these subproperties has one or two annotations, which via OWL annotation properties: `:hasBegining` and `:hasEnd`, explicitly determine the temporal boundaries of respective period in time (as `xsd:values`). Then each of these sub-properties is used to represent the relevant tensed fact:

```
:settlement_1 a :physical-object ;
    :hasLocationAt1425-1459 :location1 ;
    :hasLocationAt1460- :location2 .
```

```
:hasLocationAt1425-1459 a owl:ObjectProperty ;
    :hasBegining "1425-01-01T00:00:00"^^xsd:dateTime ;
    :hasEnd "1459-12-31T00:00:00"^^xsd:dateTime .
```

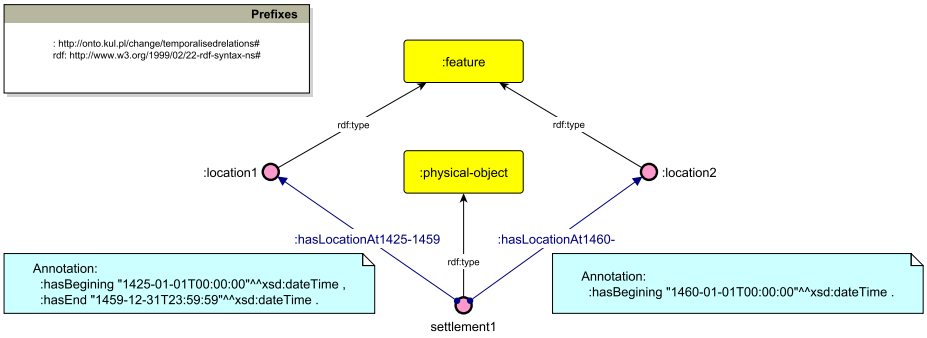


Fig. 2: Simple Pattern

4 Philosopically-Laden Patterns

If neither the standard nor the simple patterns meets your needs, we think you will have to consider a solution that involves some philosophy.

To solve the problem in question one may survey these philosophical ideas that concern how things exist in time and how they change (or are being changed). Since the topic of change is one of the perennial issues in philosophy, one needs to apply some conceptual filter – in this case we focus on the so-called analytical philosophy. In contemporary analytical philosophy *locus classicus* to look up is the debate between endurantism (aka: 3D view) and perdurantism (aka: 4D view).

Very roughly speaking (and significantly simplifying), an endurantist believes that things that exist in time occur at different moments as wholly present, i.e. each such thing has a kind of ontological core that remains its identity throughout its life – despite a number of changes the thing undergoes. So Nieszawa existed both in 1425 and 1460 because there was something, which existed as one and the same object at both times – despite the fact that Nieszawa in 1425 had a different street plan, building density, population, location, etc., than Nieszawa at 1460. Note that endurantism is, usually, interpreted in the weak sense: there are entities that occur at different moments as wholly present, but there might exists other entities that exist in time in a different way. So endurantists usually allow for such entities like processes or events, which are sometimes called perdurants, due to their specific mode of temporal existence.

Opposing to this view a perdurantist claims that *all* things that exists in time do not last (or endure) in time, so there is no such a thing as Nieszawa that

exists at two different moments in time. All things that exist in time are believed to have special kind of parts, called temporal parts, due to which they exist at different moments. So Nieszawa existed in 1425 because it has a temporal part that was present at 1425. And it existed in 1460 because it has another temporal part, which was present at 1460.

This debate may provide us with a number of solutions to this paper's problem.

4.1 Endurantist Patterns

All endurantist solutions accept endurants, i.e., things that exist in time by being wholly present at every moment they exist. The first solution we discuss is, so to speak, purely endurantistic as it does not require any other kind of entities. The temporal references are assigned here directly to endurants. The other solutions requires processes and/or events and the last two of these three assign temporal references to the latter.

Manifestation-Based Endurantist Pattern Manifestation-based endurantist pattern is based on the seminal Kit Fine's work [4] and has been introduced to the Semantic Web community during FOIS 2016 [10] by Makolab's R&D Group. The basic idea of this approach is that all entities are divided into variable entities, i.e. the entities that have different manifestations at different times, and non-variable entities that do not have different manifestations as different objects at different times. Among non-variable entities are manifestations of variable entities. The following assumptions introduced by Kit Fine and generalized by F. Moltmann in [8] have been adopted in [10]:

- Each variable entity (e.g. Nieszawa) has at least one manifestation.
- If a variable entity is manifested at a certain time by some manifestation m, then the entity exists at t, takes the location and all other properties of the manifestation at that time. In our example we will say that Nieszawa, i.e. `:settlement1`, has two manifestations:
  1. `:settlement1_manifestation1` – for the interval from 1425 to 1459;
  2. `:settlement1_manifestation2` – for the interval from 1460 to this day.

Given these assumptions, our temporal story can be expressed by means of the following turtle serialization and graph:

```
:settlement1 a oc:VariableEntity ;
    oc:hasManifestation :settlement1_manifestation1 ,
    :settlement1_manifestation2 .

:settlement1_manifestation1 a oc:VariableEntityManifestation ;
    :hasLocation :location1 ;
    :hasBeginning "1425-01-01T00:00:00"^^xsd:dateTime ;
    :hasEnd "1459-12-31T23:59:59"^^xsd:dateTime .
```

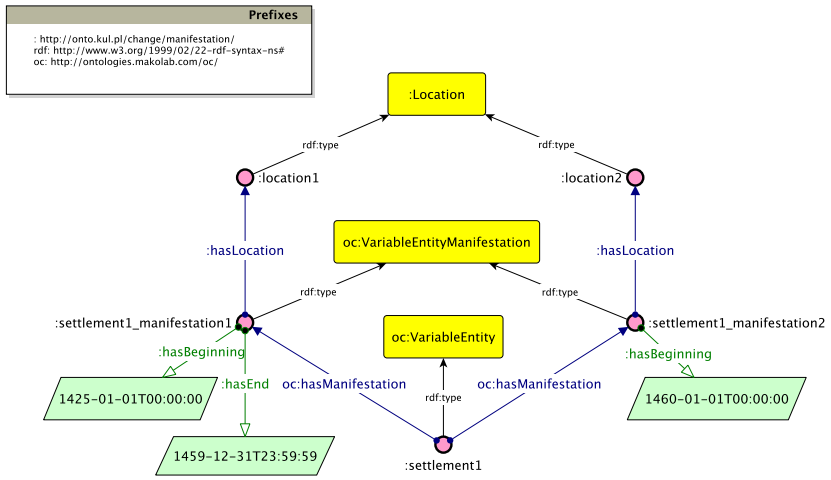


Fig. 3: Manifestation-Based Endurantist Pattern

Temporally Qualified Continuants Pattern Temporally qualified continuants (aka: TQCs), as described in [5], are to be understood by analogy to the relation between endurants (aka: continuants) and their (life) histories. Each continuant has its unique history, which contains all events and processes in which it takes part. (Technically speaking the history of an endurant is the mereological fusion of all such perdurants – see [2, p. 122-123].) And if we are interested in a particular period in its life, we can take a proper part of this history – a *phase*. Then each such phase delimits, so to speak, the endurant restricted to the phase – also known as temporally qualified continuant. Then the relations in which the original continuant is involved in time can be directly and unambiguously attached to this TQC.

```
:settlement1 a obo:BFO_0000030 ;
    obo:BFO_0000185 :settlement1_history .

:settlement1_history a obo:BFO_0000182 ;
    obo:BFO_0000117 :settlement1_phase1 ,
    :settlement1_phase2 .

:settlement1_phase1 a tqc:phase ;
    tqc:phase-of :settlement1_TQC1 .

:settlement1_TQC1 a tqc:tq-continuant ;
    obo:BFO_0000083 :location1 ;
    :hasBeginning "1425-01-01T00:00:00"^^xsd:dateTime ;
    :hasEnd "1459-12-31T23:59:59"^^xsd:dateTime .
```

State-Based Endurantist Pattern The state-based pattern needs a special kind of perdurants called states. A state is understood here in the same way as in the DOLCE ontology ([7]), i.e., as a perdurant that is cumulative and homeo-

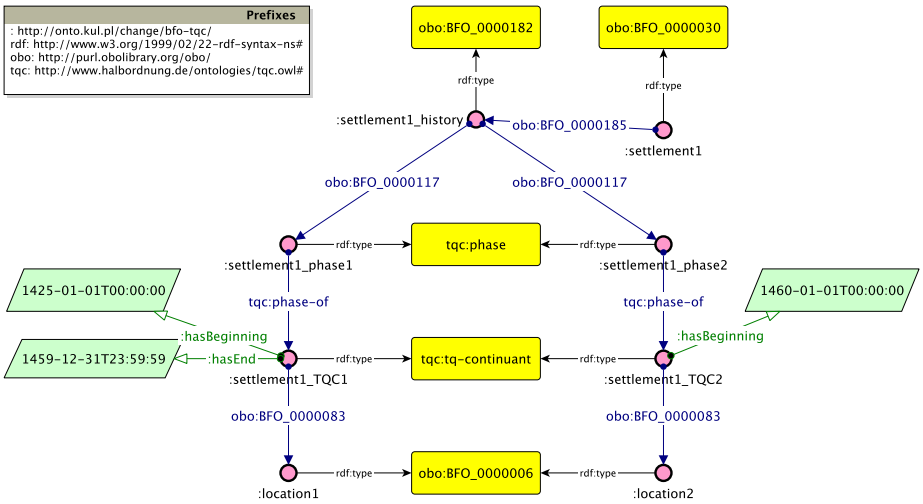


Fig. 4: Temporally Qualified Continuants Pattern

moeric. The most common example of states are such perdurants as: being connected, being sitting, being red, etc. We also need the concept of participation, by means of which we can say that an endurant participates in a perdurant, e.g., when a football player participates in a football match or a postbox participates in being red.

In order to express that a certain object has a certain property at a certain time (be it a point in time or an interval) we take the state such that:

- its temporal boundaries are determined by the time;
- both the object and the property participate in it.

For instance, to represent the fact that Nieszawa is located at location 1 from 1425 to 1459, we need, besides Nieszawa and location 1, also the state of Nieszawa being located at location 1 in 1425, (i) whose boundaries are determined by the beginning of 1425 and the end of year 1459 and (ii) such that Nieszawa and location 1 participate in this state. Obviously, a similar structure is needed for Nieszawa's location 2.

To implement this idea in an OWL ontology we will use DOLCE-Lite OWL version[3] despite the fact that it introduces some extra complexity due to the way DOLCE handles properties and their values. The reason for this design decision is based on the pattern's conceptual requirements: the concept of cumulative and homeomeric perdurants and the relation of participation, which get their intended meanings within the context of DOLCE. In other words, the pattern at stake is not just a formal hack to handle ternary relations within the binary constraints, but it requires some background ontology whose role is to provide

[3] See: `http://www.loa.istc.cnr.it/ontologies/DOLCE-Lite.owl`

the proper meanings for certain terms and predicates. We do not, however, either claim or assume that the state-based pattern we implement in DOLCE is the only one congruent with the philosophical presuppositions of this ontology. That is to say, we admit that one can implement other pattern(s) within this ontology, some of which may better harmonise with those presuppositions. On the other hand, we took the effort to guarantee that our implementation is both internally consistent and is consistent with the main specification of this ontology, as defined in [7]. The details of the pattern are shown below:

```
:settlement1 a dolce:physical-object .

:settlement1_state_location1 a dolce:state ;
    :constant-participant :location_1 ,
    :settlement1 ;
    dolce:has-quality :temporallocation_settlement1_state_location1 .

:temporallocation_settlement1_state_location1 a dolce:temporal-location_q ;
    dolce:has-quale :value_temporallocation_settlement1_state_location1 .

:value_temporallocation_settlement1_state_location1 a dolce:time-interval ;
    :hasBegining "1425-01-01T00:00:00"^^xsd:dateTime ;
    :hasEnd "1459-12-31T23:59:59"^^xsd:dateTime .
```

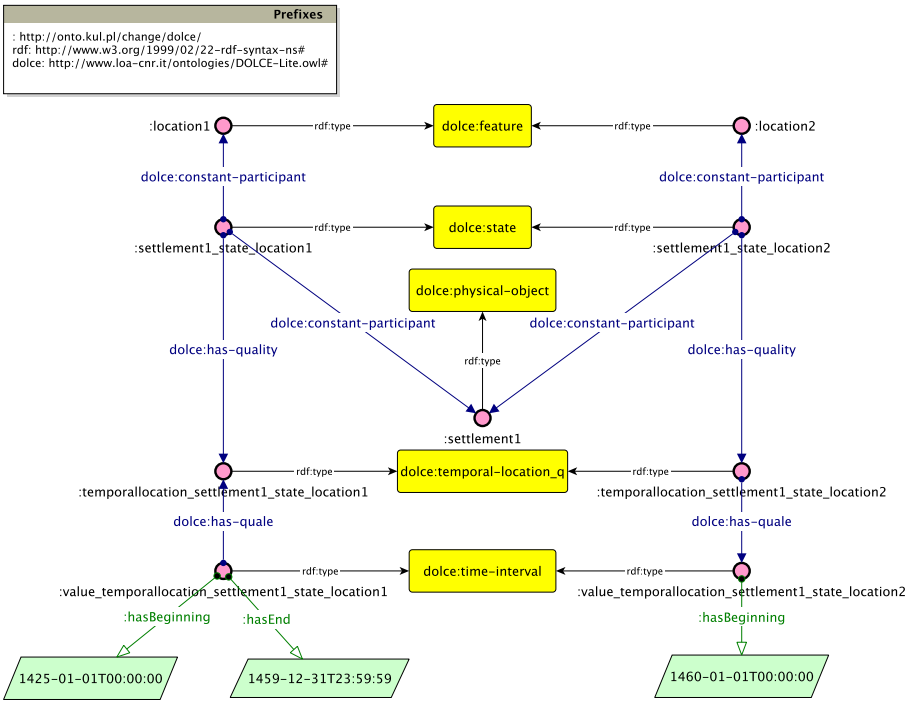


Fig. 5: State-Based Endurantist Pattern

History-Based Endurantist Pattern The history-based pattern requires the notion of endurant history, which we know from section 4.1, and the conception of mereology, i.e., formal theory of parthood relation. This time we need to signify three histories:

1. history of Nieszawa (`:settlement1_history`)
2. history of location 1 (`:location1_history`)
3. history of location 2 (`:location2_history`)

Then we will consider an individual that is the mereological intersection (or greatest common part) of the history of Nieszawa and the history of location 1, say `:product_histories_settlement_1_location_1`, and we identify the temporal boundaries of this intersection by years 1425 and 1459. Obviously, a similar structure is needed for Nieszawa's location 2.

The implementation of this pattern is embedded in BFO 2.0 ontology[4]. Again this adds some conceptual overhead to the pattern, which we are ready to accept for the same reason as in the previous case. And as before we flesh out this pattern only by means of the figure 6.

```
:settlement1 a obo:BFO_0000030 ;
    obo:BFO_0000185 :settlement1_history .

:location1 a obo:BFO_0000029 ;
    obo:BFO_0000067 :location1_history .

:product_histories_settlement1_location1 a obo:BFO_0000015 ;
    obo:BFO_0000132 :location1_history ,
    :settlement1_history ;
    :hasBegining "1425-01-01T00:00:00"^^xsd:dateTime ;
    :hasEnd "1459-12-31T23:59:59"^^xsd:dateTime .
```

4.2 Perdurantist Pattern

Let us repeat that perdurantism represents all objects that exist in time as if they were processes. So the perdurantist solution splits, so to speak, Nieszawa, which is considered here as a process, into two temporal parts (or sub-processes):

1. temporal part of Nieszawa that existed from 1425 until 1459 – we will represent it as `:temporalpart1_settlementunit1`;
2. temporal part of Nieszawa that existed from 1460 onwards – we will represent it as `:temporalpart2_settlementunit1`.

Then instead of positing Nieszawa being located in location 1 from 1425 until 1459, which would require a ternary relationship, we simply state that `:temporalpart1_settlementunit1` is located in `:location1`. And similarly for the other location of Nieszawa – use `:temporalpart2_settlementunit1`.

To make this pattern more palatable we will express it in GFO ontology[5] with the same caveats as before. Then we can express the relocation of Nieszawa as follows:

[4] `http://ifomis.uni-saarland.de/bfo/`.
[5] See: `http://www.onto-med.de/ontologies/gfo/`

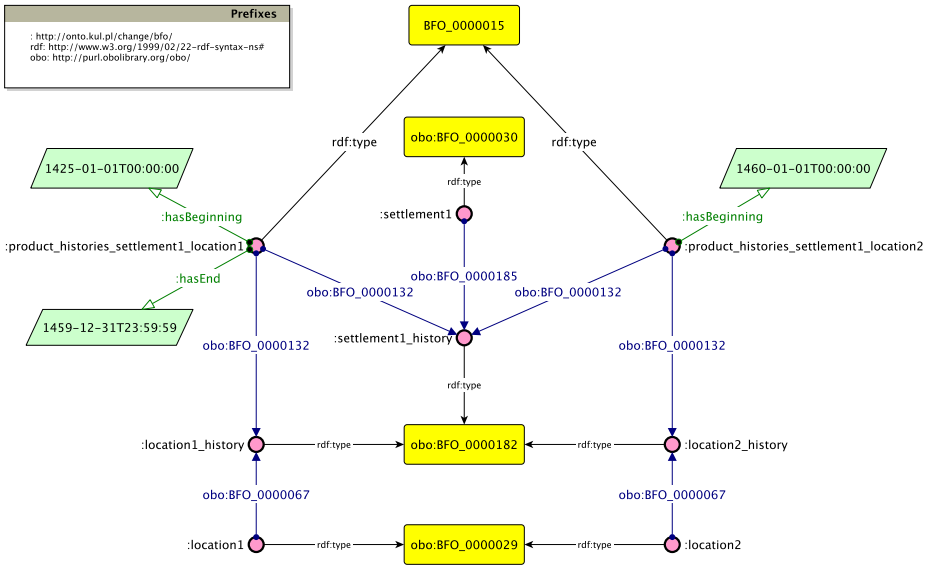


Fig. 6: History-Based Endurantist Pattern

```
:settlement1 a gfo:Perpetuant ;
    gfo:has_proper_part :temporalpart1_settlementunit1 ,
    :temporalpart2_settlementunit1 .

:temporalpart1_settlementunit1 a gfo:Material_object ;
    :hasBegining "1425-01-01T00:00:00"^^xsd:dateTime ;
    :hasEnd "1459-12-31T23:59:59"^^xsd:dateTime ;
    gfo:occupies :location1 .
```

5 Pattern Comparison

So there are at least seven (or eight if you distinguish between two reification approaches) patterns by means of which one can represent binary tensed relationships in OWL. Is there such a thing as the best pattern or the pattern to be recommended over other patterns? Although such questions are usually of little research value, we will compare in this section the above patterns to indicate their strengths and limitations.

The first comparison is actually already embedded in the main classification of patterns: there are two (or three) patterns free from any philosophical assumptions and five patterns where you need to accept a certain philosophical view to use them in the proper way. If the latter fact is considered as an unacceptable liability, then the Standard and Simple Patterns fare better in this respect than the other patterns. In passing, we note that TQC and History-Based Endurantist Patterns attest that one (top-level) ontology may be extended with more than one pattern of tensed relations.

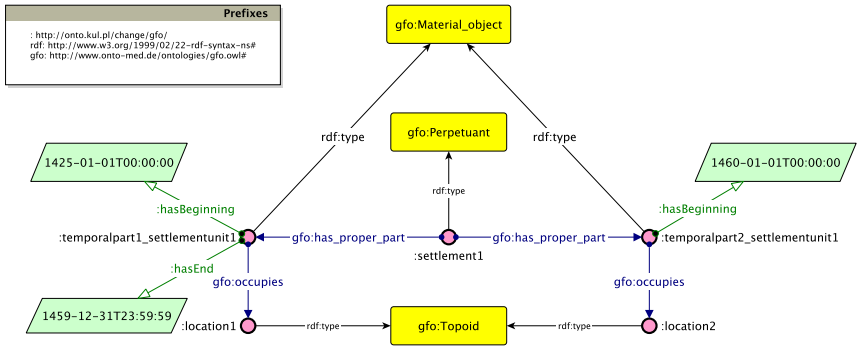


Fig. 7: Perdurantist Pattern

There are two more comparisons that can be drawn between these patterns.

5.1 Pattern Size Comparison

First, we did a simple calculation of the sizes of the respective patterns as applied to our running example – see tables 1 and 2. Given the assumption that smaller patterns are to be preferred to bigger ones, the Simple Pattern turns out to be the most economic and the most demanding are the TQC and the State-Based Endurantist Patterns. Still we should emphasise that this comparison is skewed by the particular patterns' implementations – the most acute error concerns the specific way in which DOLCE ontology renders qualities and their values. Note, by the way, that the ABoxes of Manifestation-Based Endurantist Pattern and Perdurantist Pattern are not only equinumerous but they are also isomorphic.

5.2 Pattern Usage Scenario Comparison

The second comparison may be drawn with respect to usage scenarios, which take into account some specific types of changes one may need to represent. This comparison is multi-facetted, i.e., we classify the usage scenarios with respect to two facets:

1. change type cardinality facet, which takes into account a number of simultaneous change types one needs to track down, e.g., whether one needs to represent only settlement location changes or location changes together with changes in settlement names, settlement types, etc. ;
2. arity facet, which takes into account change"arities", i.e., whether one represents changes within (monadic) categories, binary relations, ternary relations, etc.

Change Type Cardinality Facet We consider here two types of usage scenarios:

1. mono-change scenarios, where only one type of change is recorded;

Pattern	Classes	Object Properties	Datatype Properties	Annotation Properties
Standard Pattern(s)	3	2	2	0
Simple Pattern	2	2	0	2
TQC Pattern	5	4	2	0
Manifestation-Based Endurantist Pattern	3	2	2	0
State-Based Endurantist Pattern*	5	3	2	0
History-Based Endurantist Pattern*	4	2	2	0
Perdurantist Pattern	3	2	2	0

Table 1: Patterns' TBox Size Comparison

Pattern	Triples	Individuals
Standard Reification Pattern(s)	12	5
Simple Pattern	8	3
TQC Pattern	18	8
Manifestation-Based Endurantist Pattern	12	5
State-Based Endurantist Pattern*	20	9
History-Based Endurantist Pattern*	19	8
Perdurantist Pattern	12	5

Table 2: Patterns' ABox Size Comparison

* indicates that a pattern's counts are skewed by the pattern's implementation

2. multi-change scenarios, where we need to record multiple types of changes.

It seems to us that the multi-change scenarios may pose the following problem to some of the patterns we discussed. Consider, for instance, Standard Pattern(s). According to the W3C recommendation reification requires a separate reified class (similar to `:LocationRelation`) for each type of reified relation. In the context of tensed relations this probably translates to the requirement that each tensed relations gets its own reification class. Going back to our example, in order to represent the variability of Nieszawa's name (actually it did change a few times in its history), one would have to introduce `:NameRelation` and carry out a similar reification as we did for Nieszawa's location variability. As a result we will end up with two classes, one for each reified tensed relationship, i.e., one for each type of change. Then even if Nieszawa changed its location and name at the same point in time, we would end up with two instances of these classes. On the other hand, such patterns as the TQC or the Manifestation-Based Endurantist Patterns require only one individual. Consequently, they are more parsimonious that Standard Pattern(s).

This observation led us to appreciate the need for some kind of quantitative comparison of our patterns with respect to the facet in question. So to estimate their "performance" in this respect we calculated (cf. tables 3 and 4) the sizes of their TBoxes and ABoxes in a multi-change scenario with the following parameters:

– we track one changing individual;

– we track its n tensed relations – for simplicity we assume that the domain and range of each such relation is `owl:Class`;
– we record its status both at the beginnings and at the ends of t intervals;[6]
– in each case we assume "the worst-case scenario": at each interval the individual is related to different objects.

Pattern	**Classes**	**Object Properties**	**Datatype Properties**	**Annotation Properties**
Standard Pattern(s)	$n+1$	$2 \cdot n$	2	0
Simple Pattern	1	$n \cdot t$	0	2
TQC Pattern	4	$n+1$	2	0
Manifestation-Based Endurantist Pattern	3	$n+1$	2	0
State-Based Endurantist Pattern*	4	$n+3$	2	0
History-Based Endurantist Pattern*	2	$n+2$	2	0
Perdurantist Pattern	3	$n+1$	2	0

Table 3: Patterns' TBox Size Comparison In Multi-Change Scenarios

Pattern	**Triples**	**Individuals**
Standard Reification Pattern(s)	$6 \cdot n \cdot t+1$	$2 \cdot n \cdot t+1$
Simple Pattern	$3 \cdot n \cdot t+1$	$n \cdot t+1$
TQC Pattern	$8 \cdot n \cdot t+3$	$3 \cdot n \cdot t+2$
Manifestation-Based Endurantist Pattern	$(n+5) \cdot t+1$	$(n+1) \cdot 5+1$
State-Based Endurantist Pattern*	$10 \cdot n \cdot t+1$	$4 \cdot n \cdot t+1$
History-Based Endurantist Pattern*	$8 \cdot n \cdot t+2$	$3 \cdot n \cdot t+2$
Perdurantist Pattern	$(n+5) \cdot t+1$	$(n+1) \cdot 5+1$

Table 4: Patterns' ABox Size Comparison In Multi-Change Scenarios

Figure 8 visualises how the number of triples grows for Simple and Perdurantist Patterns with the number of change types and the number of intervals.

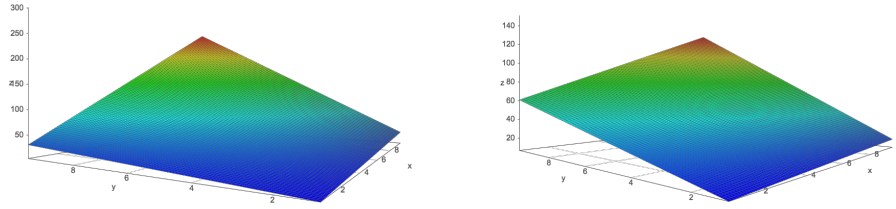
Arity Facet Here we consider three types of usage scenarios:

1. monadic scenarios, where change concerns (monadic) categories, i.e. we record changes with respect to (monadic) categories;
2. binary scenarios, where change concerns binary relations, i.e. we record changes with respect to binary relations;
3. higher-arity scenarios, where change concerns ternary, quoternary, etc. relations, i.e. we record changes with respect to higher arity relations.

First note that all patterns, except for Standard Pattern(s), do not handle higher-arity scenarios, so in this aspect the latter trumps all other patterns.

Secondly, Standard Pattern(s), TQC Pattern, Manifestation-Based Pattern, and Perdurantist Pattern are capable of representing changes with respect to

[6] Note that the running examples does not satisfy this assumption.



(a) Simple Pattern (b) Perdurantist Pattern

Fig. 8: Pattern triples growth (z dimension) with respect to number of tensed relation (x dimension) and number of intervals (y dimension)

(monadic) categories in the same way as they represent changes with respect to binary relations. The situation is different with respect to other patterns. Consider, for instance, a process due to which Nieszawa changes its type from village to town. How can one express this type of change within State-Based Endurantist Pattern or History-Based Endurantist Pattern? On the other hand, although Simple Pattern does not explicitly provide with a pattern for monadic scenario, it is possible to extend it so that it will cover tensed (monadic) categories. To this end, one needs to temporally qualify OWL classes instead of OWL object properties, so, for instance, one would need to have such classes as `Village1425`, `Village1459`, etc., as subclasses of `Village` class to contain those entities that were villages, respectively, in 1425 and in 1459.

In sum, given the usage scenarios envisaged in this classification, the most flexible pattern is the Standard Pattern(s) with the "fore-runners" of the TQC, Manifestation-Based, and Perdurantist Patterns. Simple Pattern seems to be extendable to monadic changes, but the other two patterns do not make room for any type of change besides changes with respect to binary relations.

6 Further work

As mentioned in section 1 the main rationale for this research is the need to represent time and change in a historical geoinformation system. Consequently, we see this paper as a preparatory work to provide for the crucial design choice concerning the logical schema of the database component of this system and its triple store component where the data will be made available as a LOD dataset. Then, when the choice is made, we will be able to implement the selected pattern and validate its applicability not on a single, simplified example but in large scale. This validation will concern not only the technical aspects discussed in this paper but also how the selected pattern is to facilitate the data management for the laymen users of the system.

References

1. Time Ontology in OWL (September 2017), http://www.w3.org/TR/owl-time
2. Arp, R., Smith, B., Spear, A.D.: Building Ontologies with Basic Formal Ontology. The MIT Press (2015)
3. Brogaard, B.: Tensed relations. Analysis 66, 194–202 (2006)
4. Fine, K.: Things and their parts. Midwest Studies in Philosophy 23(1), 61–74 (1999)
5. Grewe, N., Jansen, L., Smith, B.: Permanent generic relatedness and silent change. In: Kutz et al. [6], http://ceur-ws.org/Vol-1660/competition-paper1.pdf
6. Kutz, O., de Cesare, S., Hedblom, M.M., Besold, T.R., Veale, T., Gailly, F., Guizzardi, G., Lycett, M., Partridge, C., Pastor, O., Grüninger, M., Neuhaus, F., Mossakowski, T., Borgo, S., Bozzato, L., Vescovo, C.D., Homola, M., Loebe, F., Barton, A., Bourguet, J. (eds.): Proceedings of the Joint Ontology Workshops 2016 Episode 2: The French Summer of Ontology co-located with the 9th International Conference on Formal Ontology in Information Systems (FOIS 2016), Annecy, France, July 6-9, 2016, CEUR Workshop Proceedings, vol. 1660. CEUR-WS.org (2016), http://ceur-ws.org/Vol-1660
7. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A.: WonderWeb Deliverable D18. The WonderWeb Library of Foundational Ontologies and the DOLCE ontology (2003), http://wonderweb.semanticweb.org/deliverables/documents/D18.pdf
8. Moltmann, F.: Variable Objects and Truth-Making. In: Dumitru, M. (ed.) Metaphysics, Meaning, and Modality. Themes from Kit Fine, vol. (forthcoming). Oxford Univerity Press (2016)
9. Noy, N., Rector, A.: Defining N-ary Relations on the Semantic Web. W3C Working Group Note, World Wide Web Consortium (April 2006), http://www.w3.org/TR/swbp-n-aryRelations/
10. Trypuz, R., Kuzinski, D., Sopek, M.: General legal entity identifier ontology. In: Kutz et al. [6], http://ceur-ws.org/Vol-1660
11. Welty, C.A., Fikes, R.: A Reusable Ontology for Fluents in OWL. In: Bennett, B., Fellbaum, C. (eds.) FOIS. Frontiers in Artificial Intelligence and Applications, vol. 150, pp. 226–236. IOS Press (2006)