A Detailed Comparison of Seven Approaches for the Annotation of Time-Dependent Factual Knowledge in RDF and OWL

Hans-Ulrich Krieger

German Research Center for AI (DFKI GmbH)
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
krieger@dfki.de

Abstract

Representing time-dependent factual knowledge in RDF and OWL has become increasingly important in recent times. Extending OWL relation instances or RDF triples with further temporal arguments is usually realized through new individuals that hide the range arguments of the extended relation. As a result, reasoning and querying with such representations is extremely complex, expensive, and error-prone. In this paper, we discuss several well-known approaches to this problem and present their pros and cons. Three of them are compared in more detail, both on a theoretical and on a practical level. We also present schemata for translating triple-based encodings into general tuples, and vice versa. Concerning query time, our preliminary measurements have shown that a general tuple-based approach can easily outperform triple-based encodings by several orders of magnitude.

Keywords: temporal annotation; synchronic & diachronic relations; binary vs. N-ary representation schemata for factual statements.

1. Introduction

Representing temporally-changing information becomes increasingly important for reasoning and query services defined on top of RDF and OWL, for practical applications such as business intelligence in particular, and for the Semantic Web/Web 2.0 in general. Extending *binary* OWL ABox relation instances or RDF triples with further temporal arguments translates into a massive proliferation of useless "container" objects. Reasoning and querying with such representations is extremely complex, expensive, and error-prone.
In this paper, we critically discuss several well-known approaches to the encoding of time-dependent information in RDF and OWL. We present seven approaches and explain their pros and cons. Three of them are then compared in more detail, both theoretically and practically w.r.t. space consumption and answer time for simple queries. Two of the three approaches stay within the existing RDF paradigm, whereas the third proposal argues for replacing the RDF triple by a more general tuple in order to ease reasoning and querying, but also to come up with ontologies that have a smaller memory footprint when compared to semantically equivalent triple-based encodings.

In order to make the measurements for the three approaches comparable, we have used the rule-based semantic repository *HFC* (Krieger, 2013) that we have developed over the last years and which is comparable to popular engines, such as Jena, OWLIM, or Virtuoso. We also present schemata for translating temporal triple-based encodings into general tuples, and vice versa. Concerning query time, our preliminary measurements have shown that a general tuple-based approach can easily outperform a triple-based encoding by 1 to 5 orders of magnitude.

2. Synchronic and Diachronic Relations

Linguistics and philosophy make a distinction between synchronic and diachronic relations in order to characterize statements whose truth value do (or do not) change over time. *Synchronic* relations, such as dateOfBirth, are relations whose instances do not change over time, thus there is no direct need to attach a temporal extent to them. Consider, e.g., the natural language sentence

Tony Blair was born on May 6, 1953.

Assuming a RDF-based N-triple representation (Grant and Beckett, 2004), an information extraction (IE) system might yield the following set of triples:

```
tb rdf:type Person
tb hasName "Tony Blair"
tb dateOfBirth "1953-05-06"^^xsd:date
```

Since there is only *one unique* date of birth, this works perfectly well and properly capture the intended meaning.
Diachronic relationships, however, vary with time, i.e., their truth value do change over time. Representation frameworks such as OWL that are geared towards unary and binary relations can not directly be extended by a further (temporal) argument. Consider the following sentence:

Christopher Gent was Vodafone's chairman until July 2003. Later, Chris became the chairman of GlaxoSmithKline with effect from 1st January 2005.

Given this, an IE system might discover the following time-dependent facts:

```
[????-??-??,2003-07-??]: cg isChairman vf
[2005-01-01,????-??-??]: cg isChairman gsk
```

Applying the synchronic temporal representation schema from above gives us

```
cg isChairman vf
cg hasTime [????-??-??,2003-07-??]
cg isChairman gsk
cg hasTime [2005-01-01,????-??-??]
```

However, the resulting RDF graph mixes up the association between the original statements and their temporal extent

```
 [????-??-??,2003-07-??]: cg isChairman vf
*[2005-01-01,????-??-??]: cg isChairman vf
*[????-??-??,2003-07-??]: cg isChairman gsk
 [2005-01-01,????-??-??]: cg isChairman gsk
```

as the second and third association is not supported by the above natural language quotation.

3. Approaches to Diachronic Representation

Several well-known techniques of extending *binary* relations with additional arguments have been proposed in the literature.

3.1. Equip Relation With Temporal Arguments

This approach has been pursued in temporal databases (called *valid time*) and the logic programming community. For instance, a binary relation, such as worksFor between a person p of type Person and a company c of type Company becomes a quaternary relation with two further temporal arguments s and e, expressing the temporal interval $[s, e]$ in which the atemporal statement worksFor(p, c) is true (instants are represented by stating that $s = e$):

$$\mathsf{worksFor}(p, c) \;\longmapsto\; \mathsf{worksFor}(p, c, \underline{s, e})$$

Unfortunately, OWL and description logic (DL) in general only support unary (classes) and binary (properties) relations in order to guarantee decidability of the usual inference problems. Thus forward chaining engines (such as OWLIM and Jena) as well as tableaux-based reasoners (e.g., Racer or Pellet) are unable to handle such descriptions.

We note here that this approach is clearly the *silver bullet* of representing binary factual statements, since it is the easiest and most natural one, although a direct interpretation is incompatible with RDF and almost all currently available reasoners. We will favor this kind of representation in the second part of the paper when presenting the measurements, using *HFC* (Krieger, 2013).

3.2. Apply a Meta-Logical Predicate

McCarthy & Hayes' situation calculus, James Allen's interval logic, and the knowledge representation formalism KIF use variants of the meta-logical predicate holds. Hence, our worksFor(p, c) relation instance becomes holds(worksFor$(p, c), t)$. McCarthy & Hayes call a statement whose truth value changes over time a *fluent* (McCarthy and Hayes, 1969). The extended quaternary relation from the previous subsection can be seen as a *relational* fluent, whereas the holds expression here, however, embodies a *functional* fluent, meaning that worksFor(p, c) is assumed to yield a situation-dependent value.

Such kinds of relations are *not* possible in OWL, since description logics limit themselves to subsets of *function-free* first-order logic and because only a weak form of relation composition is possible in OWL. However, we can reify the atemporal fact worksFor(p, c) in RDF, so that the above holds relation instance can at least be *encoded* by introducing a new individual o, represented as an RDF blank node. We note that in the original calculus, situations were defined at an *instant* of time, thus we use only a single temporal argument t here.

$$\underline{\mathsf{holds}}(\mathsf{worksFor}(p, c), \underline{t}) \;\longmapsto\; \exists o \,.\, \mathsf{holds}(o, t) \wedge \\ \mathsf{type}(o, \mathsf{AtemporalFact}) \wedge \mathsf{subject}(o, p) \wedge \\ \mathsf{predicate}(o, \mathsf{worksFor}) \wedge \mathsf{object}(o, c)$$

As an alternative, we might turn the worksFor relation into a class:

$$\underline{\mathsf{holds}}(\mathsf{worksFor}(p, c), \underline{t}) \;\longmapsto\; \exists o \,.\, \mathsf{holds}(o, t) \wedge \\ \mathsf{type}(o, \mathsf{WorksFor}) \wedge \mathsf{subject}(o, p) \wedge \mathsf{object}(o, c)$$

However, this would require to always introduce a new class for the representation of each diachronic relation.

3.3. Reify the Original Relation

Reifying a relation instance again leads to the introduction of a new object and five additional new relationships. In addition, a new class needs to be introduced for *each* reified relation, plus accessors to the original arguments, very similar to the approach directly above. Furthermore, and very important, relation reification loses the original relation name, thus requiring a massive modification of the original ontology.

Coming back to our worksFor example, we obtain (WorksFor is the newly introduced class)

$$\underline{\mathsf{worksFor}(p, c, s, e)} \;\longmapsto\; \exists o \,.\, \mathsf{type}(o, \mathsf{WorksFor}) \wedge \\ \mathsf{person}(o, p) \wedge \mathsf{company}(o, c) \wedge \\ \mathsf{starts}(o, s) \wedge \mathsf{ends}(o, e)$$

It is worth noting that this encoding can be seen as a kind of "owlfication" of *Neo-Davidsonian* semantics (Parsons, 1990), as the original relation is turned into an *event*.

3.4. YAGO's Fact Identifier

The approach YAGO (Hoffart et al., 2011) takes is related to Approach 2 and 3 directly above, as it is a kind of *external* reification. YAGO uses its own extension of the N3 plain triple format, called N4, which associate unique identifiers i with each time-dependent fact.

The above quaternary relation instance then is represented as follows:

$$\underline{\mathsf{worksFor}(p, c, s, e)} \;\longmapsto\; \exists i \,.\, i : \mathsf{worksFor}(p, c) \wedge \\ \mathsf{occursSince}(i, s) \wedge \mathsf{occursUntil}(i, e)$$

Note that the association i : worksFor(p, c) has the disadvantage of *not* being part of the triple repository (as it is a quadruple technically; we guess that there exists a separate extendable mapping table). Thus, entailment rules and queries will never have access to these quadruples, unless some custom functionality has been implemented in the semantic repository. Nevertheless, this is a valid and proper *annotation* schema, however *not* expressible in OWL.

Rather, such a kind of association can be seen as an extension of the idea behind *annotation properties* in OWL in that not only classes, properties, and individuals can be annotated with information, but also binary relation instances (= triples), thus occursSince and occursUntil from above can be regarded as relation instance annotation properties. Unfortunately, we are not aware of such an extension.

3.5. Wrap Range Arguments

Wrapping the range arguments of a relation instance, i.e., grouping them in a new object, allows us to keep the original relation name, although the approach still requires to rewrite the original ontology:

$$\begin{aligned}&\mathsf{worksFor}(p, \underline{c, s, e}) \;\longmapsto\; \exists o \,.\, \mathsf{worksFor}(p, o) \wedge \\ &\quad \mathsf{type}(o, \mathsf{CompanyTime}) \wedge \mathsf{company}(o, c) \wedge \\ &\quad \mathsf{starts}(o, s) \wedge \mathsf{ends}(o, e)\end{aligned}$$

Again, a new object (o), a new class (CompanyTime), and new accessors (company, starts, ends) need to be introduced. W3C suggests this obvious pattern to be used to encode arbitrary *N-ary relations* (Hayes and Welty, 2006). Alternatively, instead of defining a new class for each range type of the original relation, one might define a general class, say RangePlusTime, together with three accessors value, starts, and ends, in order to avoid a *reduplication* of the original class hierarchy on the property level. We use the latter refinement in our measurements below.

3.6. Encode the 4D View in OWL

(Welty and Fikes, 2006) have presented an implementation of the 4D or perdurantist view in OWL, using so-called time slices (Sider, 2001). Relations from the original ontology no longer connect the original entities, but instead connect time slices that belong to those entities. A time slice here is merely a container for storing the time dimension of spacetime. At least, the original relation name is kept, although such a representation requires a lot of rewriting and even introduces *two* new container objects:

$$\begin{aligned}&\underline{\mathsf{worksFor}(p, c, s, e)} \;\longmapsto\; \exists t, t' \,.\, \mathsf{worksFor}(t, t') \wedge \\ &\quad \mathsf{type}(t, \mathsf{TimeSlice}) \wedge \mathsf{hasTimeSlice}(p, t) \\ &\quad \mathsf{type}(t', \mathsf{TimeSlice}) \wedge \mathsf{hasTimeSlice}(c, t') \\ &\quad \mathsf{starts}(t, s) \wedge \mathsf{ends}(t, e) \wedge \\ &\quad \mathsf{starts}(t', s) \wedge \mathsf{ends}(t', e)\end{aligned}$$

We note here that *this* approach and the approach *below* only work for binary relations. This restriction, however, do no harm to RDF-encoded OWL ontologies, since an RDF triple encodes a binary relation.

3.7. Interpret Original Entities as Time Slices

In (Krieger et al., 2008), we have slightly extended and at the same time simplified the perdurantist/4D view from directly above. p and c from the example above are still first-class citizens, now called *perdurants* which possess time slices, *explaining* the behavior of an entity within a certain temporal extent (e.g., being a Person or a Company) and are able to group multiple facts that stay *constant* within the same period of time. In the extended relation instance, p and c are then replaced by new IDs p' and c' (similar to the approach above), but these new individuals are still typed to the original classes, here: Person and Company, resp. Keeping the original typing thus allows us to superimpose the original class hierarchy with the notion of a time slice.

$$\begin{aligned}&\underline{\mathsf{worksFor}(p, c, s, e)} \;\longmapsto\; \exists p', c' \,.\, \mathsf{worksFor}(p', c') \wedge \\ &\quad \mathsf{type}(p', \mathsf{Person}) \wedge \mathsf{hasTimeSlice}(p, p') \\ &\quad \mathsf{type}(c', \mathsf{Company}) \wedge \mathsf{hasTimeSlice}(c, c') \\ &\quad \mathsf{starts}(p', s) \wedge \mathsf{ends}(p', e) \wedge \\ &\quad \mathsf{starts}(c', s) \wedge \mathsf{ends}(c', e)\end{aligned}$$

The nice thing with this reinterpretation is that it does not require any rewriting of the TBox and RBox of an ontology and makes it easy to equip arbitrary upper and domain ontologies with a concept of time, supplied by an independent time ontology (e.g., OWL-Time) that only needs to talk about instants and/or intervals; see (Krieger, 2010).

Perdurants p and c above only need to be introduced once, independent of which time slice they are linked to. For example, assuming perdurant p possesses three time slices for worksFor(p', c', s, e), worksFor(p'', c'', s, e), and hasWorkAddress(p''', a', s, e). Since the starting and ending time coincide in the three statements, p', p'', and p''' can be identified, and the temporal extent needs to be specified only once (and not three times).

4. Theoretical Considerations

Within this section, we will consider three of the above seven approaches (Sections 3.1.–3.7.) which we find to be the most promising ones. On a theoretical level, we will count how many bytes, tuple elements, and triples/tuples overall are needed to represent a diachronic relation instance, using approaches **1**, **5**, and **7**.

During the last years, we have gained some experience with all three formats in several German and European projects. In the European project *NIFTi* and *TrendMiner*, we have applied Approach 1 (Krieger and Kruijff, 2011; Krieger and Declerck, 2014). The German *TAKE* project has used Approach 5 to store biographical knowledge. The ontology which backs up the LT-World language portal had been rewritten to adhere to Approach 5, as it lacked an explicit treatment of time. In *MUSING*, we have used Approach 7 to equip the PROTON upper ontology with a notion of time (Leibold et al., 2010). For the *MONNET* project, we have also chosen Approach 7 to represent the Web content of companies, listed on Deutsche Börse's DAX and NYSE's Euronext.

In the following, we will restrict ourself to quaternary relations $p \subseteq D \times R \times T \times T$, where T is used to describe the starting and ending point of a fluent. The reason for this is that approach 7 (and 6) only works for binary relations that are extended by one or two further temporal arguments. Thus a quaternary diachronic relation instance $p(d, r, s, e)$ encodes a truth value for $p(d, r)$ within interval $[s, e]$. We are neutral as to whether temporal intervals are convex (i.e., contain "holes") or whether the temporal metric utilizes $\mathbb{N}$, $\mathbb{Q}$, or $\mathbb{R}$ for T—this is unimportant for the presentation above and the measurements below. We finally note that T can be easily extended by a further disjoint element, say ?, in order to permit left-open or right-open temporal intervals. Given this, comparison operators over time instants or the Allen relations over intervals, however, no longer will be Boolean, but instead become three-valued relations.

4.1. Approach 1: Quintuples

The quaternary relation instance $p(d, r, s, e)$ is represented as a tuple in *HFC* by an extension of the plain N-triple format (Grant and Beckett, 2004):

```
  d p r s e
```

This tuple consists of 5 elements/arguments and requires (at least) 20 (= $5 * 4$) bytes, assuming an (internal) `int[]` representation with 4 byte integers (which is the case in *HFC*). Using integer arrays is a common way to represent triples/tuples internally, since the external representation of URIs and XSD atoms needs to be addressed only during input and output. Overall, we obtain 1 object (the integer array) to represent the whole tuple. This last number is very important, since it is desirable to access information directly in a semantic repository, instead of "fiddling" around with helper structures (container objects) that blow up the memory. In addition, the overall number of elements is equally important, since triple repositories usually build up large index structures to efficiently access all those triples that match a specific element at a certain position in a triple.

4.2. Approach 5: W3C's N-ary Relations

As we have indicated in Section 3.5., the triple representation of the quaternary relation instance results in 5 triples/complex objects:

```
  d p o
  o rdf:type nary:RangePlusTime
  o nary:value r
  o nary:starts s
  o nary:ends e
```

Overall, 5 triples translate into 15 (= $5 * 3$) elements or 60 (= $5 * 12$) bytes. Furthermore, for each `p`, we might need an additional class for the type of `o`, as well as accessors `value`, `starts`, and `ends`. Since these tuples need to be specified only once, we do not count them here. This approach introduces **one** brand-new individual `o` (a blank node) which turns out to be *problematic*, since it might lead to a *non-terminating closure computation* during the application of entailment rules; not covered here, see (Krieger, 2012).

4.3. Approach 7: Time Slices

As described in Section 3.7., perdurants `d` and `r` need only be introduced once, so we do not take them into account. As is the case for approach 5 above, new individuals `d′` and `r′` are introduced here; in fact, **two** for each fluent we like to represent:

```
  d′ p r′
  d′ rdf:type ...  ;; domain/range of the
  r′ rdf:type ...  ;; original relation p
  d′ fourd:starts s
  d′ fourd:ends e
  r′ fourd:starts s
  r′ fourd:ends e
  d fourd:hasTimeSlice d′
  r fourd:hasTimeSlice r′
```

This representation utilizes 9 triples, leading to 27 elements or 108 bytes per fluent in the worst case. We note here that `r′` only needs to be equipped with a temporal extent and linked to perdurant `r` iff p is an OWL *object* property, i.e., *not* mapping to XSD atoms (best case: 5 triples). The below measurements assume the *worst* case.

4.4. Comparison: When to Apply Which Approach

Let us now summarize the pros and cons of the three approaches.

Approach 1. This is—for us—the most intuitive approach: ABox relation instances are simply extended by two further temporal arguments. Existing ontologies (TBox and RBox) can be easily equipped with a treatment of time. RDFS/OWL entailment rules as well as custom rules are more intuitive, easier to formulate, and less error-prone when compared to approach 5 and 7. Approach 1 performs best in terms of memory consumption and querying/reasoning time. Contrary to approach 5 and 7, it does *not* introduces new individuals, a precondition for guaranteeing the termination of the materialization process; see (Krieger, 2012).

Approach 5. This approach, recommended by the best practice group of W3C, is able to encode arbitrary n-ary relations (as is trivially the case for approach 1). The encoding is worth to consider if ontologies are defined from scratch and require time-dependent relations. Contrary to approach 1, approach 5 is compliant with the triple model of RDF. Unfortunately, standard RDFS and OWL reasoning is no longer possible which is also the case for approach 7. This approach introduces a new blank node for each ABox relation instance.

Approach 7. This treatment is great if an ontology is already given, but misses a notion of time. The approach does not require to rewrite the TBox and the RBox of an ontology (contrary to approach 5) and also stays inside RDF. The *time slices are possessed by perdurants* view is attractive, but is the worst of the three approaches in terms of memory consumption. Two further individuals are introduced here.

5. Practical Measurements

In order to compare the three approaches on a practical level, we need a semantic repository that is able to *directly* encode arbitrary n-ary relations (in our special case: quintuples). Popular engines, such as RACER, Pellet, Jena, OWLIM, or Virtuoso which are geared towards binary relations/RDF triples can thus *not* be applied here. As mentioned in Section 1., the experiments were performed using *HFC*, a forward chaining engine and semantic repository that we have developed over the last years and that is used in our lab.

5.1. Initial Numbers

The numbers below are computed against the mid-size ontology that backs up an earlier version of the LT-World

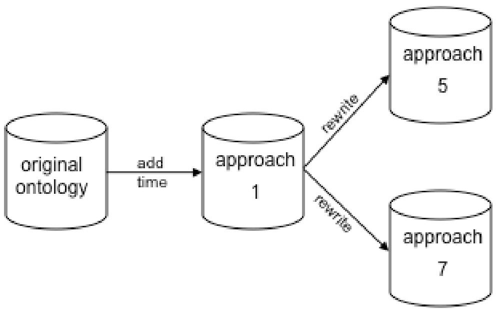

Figure 1: Rewrite schema for obtaining data sets for approaches 1, 5, and 7.

	size [MB]	#tuples	RAM [GB]	time [s]
1	53	548,132	0.42	4.3
5	129	2,740,660	1.67	14.3
7	273	4,360,428	2.15	25.9

Figure 2: Initial numbers for approaches 1, 5, and 7.

language portal (`www.lt-world.org`). The measurements are obtained on a 64bit Intel Core i7 (2.8 GHz), using Java 1.6 with an initial heap of 4GB. The unexpanded ABox consists of 204,959 RDF triples. Fully materialized, 548,132 triples are obtained. Since temporal information is missing, we randomly attach temporal starting and ending points to ABox relation instances through XSD `int` atoms which we let vary between 0 and 1,000 using a random generator (implemented by `java.lang.Math.random()`). This synthetical data (*without* the original triples) is used for approach 1.

We have then produced two further meaning-preserving data sets by rewriting the quintuples to RDF triples, compliant with the formats that are used in approach 5 and 7 (see Figure 1).

For approach 5, we have used blank nodes of type RangePlusTime to group the original value and the starting and ending time of each ABox relation instance. To address approach 7 properly, we have chosen the subject and object URIs of the original triples as names for the perdurants and have attached ascending integers to the original names in order to generate new URIs for the time slices themselves.

Given apporach 1, 5, and 7, Figure 2 then describes the three ontologies in terms of space (file size, number of triples/quintuples, main memory requirement) and loading time in order to set up *HFC* as a repository on which queries are carried out, as described in the next section.

Given these "offline" numbers, approach 1 seems to be far superior. The next section amplifies this judgment through further numbers obtained from "online" measurements for relatively easy queries.

5.2. Querying the Ontologies

This section presents measurements for six SPARQL-like queries posted in *HFC*, given approach 1, 5, and 7. The queries were originally written for approach 1 (see Figure 3) and were transformed manually to the format required by approach 5 (see Figure 4) and 7. **No** translation is depicted here for approach 7 (this would require a further half page).

The first and second query obtains the starting as well as the starting and ending times over all fluents. Query three selects those objects whose fluents are true intervals (filter: start $\neq$ end). The next query searches for subjects in symmetric relation instances that might differ in their starting and ending time. Query five simply accesses all time-stamped information for a specific individual (here: `ltw:obj_68081`). Finally, query six finds those subjects that have an ending time equal to a specific instant (here: `936`).

As can be seen in Figure 4, the queries for approach 5 (as is the case for approach 7) are no longer easy to read and take much longer to complete; in some cases this divergency *can make a difference* between doable and intractable applications which employ such kind of queries.

5.3. Comparison

As can be easily recognized from the measurements depicted in Figure 5, approach 1 easily outperforms approach 5 and 7 by **1 to 5 orders of magnitude**.

We are not only convinced that querying is faster, intuitive and less error-prone for approach 1, but have shown in (Krieger, 2012) that the same happens, even drastically for a more complex case, viz., reasoning over a temporal extension of the RDFS and OWL entailment rules (Hayes, 2004; ter Horst, 2005).

6. Summary

We hope to have shown that a general tuple-based approach for annotating time-dependent factual knowledge on the Web is far superior to triple-based approaches. We are convinced that the time is ripe to move towards this conservative extension of the RDF data model. We note here that even ontologies that utilize approaches 2 to 7 can be easily rewritten to format 1. Due to space requirements, neither are we able to depict and explain any temporal RDFS and OWL entailment rules (Krieger, 2012), nor complex custom rules in the different formats. We are certain that a closer comparison of such rules would even amplify our position, since Semantic Technologies not only are interested in accessing already externalized information (this paper), but also require inferential capabilities to make implicit knowledge explicit.

The attentive reader of this paper might ask him-/herself how we address instantiations of the above schemata in a different *external* representation format, such as XML, and how we handle relations with more than two arguments. We will speculate about this in the next two addenda.

7. Addendum 1: XML Representation

In order to use harvested data from the Web outside the RDF universe and a specific reasoner (in our case: *HFC*), it might be interesting to have an XML exchange representation for the above approaches. Unfortunately, due to the additional degree of freedom in XML to specify a value,

```
(1) SELECT DISTINCT ?start
      WHERE ?subj ?pred ?obj ?start ?end
(2) SELECT DISTINCT ?start ?end
      WHERE ?subj ?pred ?obj ?start ?end
(3) SELECT ?obj
      WHERE ?subj ?pred ?obj ?start ?end
      FILTER ?start != ?end
(4) SELECT DISTINCT ?subj
      WHERE ?subj ?pred ?obj ?start1 ?end1 &
            ?obj ?pred ?subj ?start2 ?end2
(5) SELECT *
      WHERE ltw:obj_68081 ?pred ?obj ?start ?end
(6) SELECT DISTINCT ?subj
      WHERE ?subj ?pred ?obj ?start "936"^^xsd:int
```

Figure 3: Queries for approach 1 (quintuples).

```
(1) SELECT DISTINCT ?start
      WHERE ?blank rdf:type nary:RangePlusTime &
            ?blank nary:starts ?start
(2) SELECT DISTINCT ?start ?end
      WHERE ?blank rdf:type nary:RangePlusTime &
            ?blank nary:starts ?start &
            ?blank nary:ends ?end
(3) SELECT ?obj
      WHERE ?subj ?pred ?blank &
            ?blank rdf:type nary:RangePlusTime &
            ?blank nary:value ?obj &
            ?blank nary:starts ?start &
            ?blank nary:ends ?end
      FILTER ?start != ?end
(4) SELECT DISTINCT ?subj
      WHERE ?subj ?pred ?blank1 &
            ?blank1 rdf:type nary:RangePlusTime &
            ?blank1 nary:value ?obj &
            ?obj ?pred ?blank2 &
            ?blank2 rdf:type nary:RangePlusTime &
            ?blank2 nary:value ?subj
(5) SELECT ?pred ?obj ?start ?end     ;; '*' would also show up ?blank
      WHERE ltw:obj_68081 ?pred ?blank &
            ?blank rdf:type nary:RangePlusTime &
            ?blank nary:value ?obj &
            ?blank nary:starts ?start &
            ?blank nary:ends ?end
(6) SELECT DISTINCT ?subj
      WHERE ?subj ?pred ?blank &
            ?blank rdf:type nary:RangePlusTime &
            ?blank nary:ends "936"^^xsd:int
```

Figure 4: Queries for approach 5 (W3C's N-ary relation encoding).

query [sec]	**1** (1,001)	**2** (293,880)	**3** (544,115)	**4** (1,585)	**5** (37)	**6** (1,398)
1	0.332	0.470	0.440	1.993	0.011	0.037
5	1.975	2.324	5.977	11.066	168.814	329.980
7	3.306	4.076	10.052	——	728.242	284.730

Figure 5: Processing time for the three approaches w.r.t. queries 1–6. The numbers in parentheses at the head of the table list how many results are returned by each query. Query 4 for approach 7 runs out of memory (4GB) after 96 seconds. Queries 5 and 6 are performed 100 times to measure total time.

even more kinds of representations are possible here (examples are related to approach 1 and 3, given our running worksFor example):

```
(1) <worksFor person="p" company="c" ...>
    </worksFor>

(2) <worksFor>p c s e</worksFor>

(3) <RelationInstance pred="worksFor">
      p c s e
    </RelationInstance>

(4) <Event type="worksFor">
      <person>p</person>
      <company>c</company>
      ...
    </Event>

(5) <WorksFor>
      <person>p</person>
      <company>c</company>
      ...
    </WorksFor>
```

We take a liberal stance here as our interest is not in defining an "external" exchange format, but in deciding which "internal" format performs best in terms of (i) *memory consumption*, (ii) *running time* (querying and reasoning), and (iii) *human readability*. Nevertheless, we would probably opt for either the "external" solution (4) or (5) which are related to the "internal" approach (3).

8. Addendum 2: Beyond Binary Relations

The approaches above were investigated on how well they perform w.r.t. *binary* relations whose two arguments can be considered to be *obligatory*. Such kind of relations are the default case in today's popular knowledge resources, such as YAGO, DBpedia, BabelNet, or Google's Knowledge Graph.

In case more and especially *optional* arguments are investigated, our verdict concerning the different approaches will probably turn into a different direction, so the representation format needs to be updated (in the best case) or changed (in the worst case). Consider the following example, taken from (Davidson, 1967, p. 83)

Jones buttered the toast $\underline{\text{in the bathroom}}$ $\underline{\text{with a knife}}$ $\underline{\text{at midnight}}$.

The binary base relation butter (we assume a direct mapping of the transitive verb to the relation name here) now needs to be split and/or extended by further optional arguments, as the following sentences are perfectly legal:

Jones buttered the toast.
Jones buttered the toast $\underline{\text{in the bathroom}}$.
Jones buttered the toast $\underline{\text{with a knife}}$.
Jones buttered the toast $\underline{\text{at midnight}}$.
Jones buttered the toast $\underline{\text{in the bathroom}}$ $\underline{\text{with a knife}}$.
Jones buttered the toast $\underline{\text{with a knife}}$ $\underline{\text{in the bathroom}}$.
Jones buttered the toast $\underline{\text{in the bathroom}}$ $\underline{\text{at midnight}}$.
.....

In principle, the number of adjuncts is not bounded, thus adding a large number of potentially underspecified direct relation arguments is probably a bad solution. Today's technologies often address such "hidden" arguments through a kind of *relation composition*, viz., defining further properties such as instrument (to access *knife*) or location (to access *bathroom*) on the object (*toast*) of the relation instance:

$\mathsf{instrument} \circ \mathsf{butter}$
$\mathsf{location} \circ \mathsf{butter}$

We think that modeling the optional arguments in such a way is *unsatisfactory* as instrument or location "operate" on the object of the binary relation instance and *not* on the relation instance itself!

Our *personal* solution would model the *obligatory* arguments, including (under- or unspecified) time and perhaps space, as *direct* arguments of the corresponding relation instance or tuple. A further argument, an *event* identifier, also takes part in the relation. Optional arguments, however, would be addressed through binary relations, now working on the event argument. Applying this kind of *Davidsonian* or *event* representation to the above example gives us (informal relational notation)

$$\exists e \,.\, \mathsf{butter}(e, \mathit{Jones}, \mathit{toast}, \mathit{at\ midnight}) \wedge \\ \mathsf{location}(e, \mathit{bathroom}) \wedge \\ \mathsf{instrument}(e, \mathit{knife})$$

9. Acknowledgements

The research described in this paper has been financed by the European project *TrendMiner* under contract number FP7 ICT 287863. The author has profited from discussions with Thierry Declerck and Bernd Kiefer—thank you guys! Finally, I would like to thank the three reviewers for suggestions and support.

10. References

Davidson, Donald. (1967). The logical form of action sentences. In Rescher, Nicholas, editor, *The Logic of Decision and Action*, pages 81–95. University of Pittsburgh Press.

Grant, Jan and Beckett, Dave. (2004). RDF test cases. Technical report, W3C, 10 February.

Hayes, Patrick and Welty, Chris. (2006). Defining N-ary relations on the semantic web. Technical report, W3C.

Hayes, Patrick. (2004). RDF semantics. Technical report, W3C.

Hoffart, Johannes, Suchanek, Fabian M. Berberich, Klaus, Kelham, Edwin Lewis, de Melo, Gerard, and Weikum, Gerhard. (2011). YAGO2: Exploring and querying world knowledge in time, space, context, and many languages. In *Proceedings of the 20th International World Wide Web Conference (WWW 2011)*, pages 229–232.

Krieger, Hans-Ulrich and Declerck, Thierry. (2014). TMO—the federated ontology of the TrendMiner project. In *Proceedings of the 9th edition of the Language Resources and Evaluation Conference (LREC)*.

Krieger, Hans-Ulrich and Kruijff, Geert-Jan M.˙ (2011). Combining uncertainty and description logic rule-based reasoning in situation-aware robots. In *Proceedings of the AAAI 2011 Spring Symposium "Logical Formalizations of Commonsense Reasoning"*.

Krieger, Hans-Ulrich, Kiefer, Bernd, and Declerck, Thierry. (2008). A framework for temporal representation and reasoning in business intelligence applications. In *AAAI 2008 Spring Symposium on* AI Meets Business Rules and Process Management, pages 59–70. AAAI.

Krieger, Hans-Ulrich. (2010). A general methodology for equipping ontologies with time. In *Proceedings LREC 2010*.

Krieger, Hans-Ulrich. (2012). A temporal extension of the Hayes/ter Horst entailment rules and an alternative to W3C's n-ary relations. In *Proceedings of the 7th International Conference on Formal Ontology in Information Systems (FOIS 2012)*, pages 323–336.

Krieger, Hans-Ulrich. (2013). An efficient implementation of equivalence relations in OWL via rule and query rewriting. In *Proceedings of the 7th IEEE International Conference on Semantic Computing (ICSC)*, pages 260–263.

Leibold, Christian, Krieger, Hans-Ulrich, and Spies, Marcus. (2010). Ontology-based modelling and reasoning in operational risks. In Kenett, Ron S. and Raanan, Yossi, editors, *Operational Risk Management: A Practical Approach to Intelligent Data Analysis*, chapter 3, pages 41–59. Wiley.

McCarthy, John and Hayes, Patrick J.˙ (1969). Some philosophical problems from the standpoint of artificial intelligence. In Meltzer, B. and Michie, D. editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press.

Parsons, Terence. (1990). *Events in the Semantics of English. A Study in Subatomic Semantics*. MIT Press, Cambridge, MA.

Sider, Theodore. (2001). *Four Dimensionalism. An Ontology of Persistence and Time*. Oxford University Press.

ter Horst, Herman J.˙ (2005). Combining RDF and part of OWL with rules: Semantics, decidability, complexity. In *Proceedings of the International Semantic Web Conference*, pages 668–684.

Welty, Christopher and Fikes, Richard. (2006). A reusable ontology for fluents in OWL. In *Proceedings of 4th FOIS*, pages 226–236.