Temporal Extensions to RDF

Hsien-Tseng Wang[1] and Abdullah Uz Tansel[1,2,3]

[1]*Department of Computer Science, The Graduate Center, The City University of New York, USA*
[2]*Paul H. Chook Department of Information Systems and Statistics, Baruch College, The City University of New York, USA*
[3]*School of Engineering, Thammasat University Thailand**
E-mail: hwang3@gradcenter.cuny.edu; abdullah.tansel@baruch.cuny.edu
**On leave from Baruch College, CUNY, USA*

Received 02 January 2019;
Accepted 07 March 2019

Abstract

The Semantic Web aims at building a foundation of semantic-based data models and languages for not only manipulating data and knowledge, but also in decision making by machines. Naturally, time-varying data and knowledge are required in Semantic Web applications to incorporate time and further reason about it. However, the original specifications of RDF and OWL do not include constructs for handling time-varying data and knowledge. For simplicity, RDF model is confined to binary predicates, hence some form of reification is needed to represent higher-arity predicates. To this date, there are many proposals extending RDF and OWL for handling temporal data and knowledge. They all focus on the valid time. In this paper, we examine each of these proposals and develop a taxonomy to classify them according to the form of reification employed: explicit reification or implicit reification. The implicit reification proposals are further divided into three sub-categories according to semantic constructs

Journal of Web Engineering, Vol. 18_1-3, 125–168.
doi: 10.13052/jwe1540-9589.18134
© 2019 River Publishers

they use. Some of these proposals stay compliant to the RDF and OWL standards whereas others add new constructs to RDF model and SPARQL query language. Additionally, we compare these proposed models with respect to characteristics, such as their syntax and semantics, their compliance to RDF and OWL specifications, their need for additional objects, etc. The comparison provides a useful guideline for the researchers and practitioners of the Semantic Web in managing temporal data and knowledge.

Keywords: The Semantic Web, Resource Description Framework, Taxonomy, Temporal Data, Temporal Knowledge.

1 Introduction

The Semantic Web advocated by the World Wide Web Consortium (W3C) is based on the vision of machine understandable web infrastructure and contents. In the current World Wide Web environment, web contents are constructed mainly for the presentation of data items. The term *web resource* is used to designate all kinds of web contents. Web resources are mostly consumed by human users. The Semantic Web provides additional metadata specifications, so that all identifiable resources can be annotated with metadata. The metadata layer yields the core of a semantic-based data model that facilitates description of every identifiable resource by named properties. As a result, web resources can be consumed by both human and computational agents.

The efforts led by W3C helped popularize ontology, knowledge representation and reasoning for machine processing. In Computer Science, ontology is a model of concepts and relationships among them. In this respect, an ontology is the conceptualization used to help programs, machines and humans use and share knowledge [20]. An ontological approach encodes knowledge about the world in terms of concepts, classes, instances and relationships. Its specification is materialized by using some ontology framework, such as RDF or its variants. The objective of using ontology is to create formal vocabularies, terminologies and semantic structures for using and exchanging knowledge about a domain of interest. Moreover, an inference engine, such as Pellet [45],

FaCT++ [54], etc., can be used to derive knowledge that can be logically inferred from an ontology specification.

Temporality is a common aspect of data models of all kinds and it involves changing contents over time when both the old and new contents are critical. There are two common time dimensions used in temporal databases: Valid Time and Transaction Time. Valid Time refers to the validity period of a fact, whereas Transaction Time refers to the time when that fact is recorded in the database. There is a long history of research in temporal databases as extensions of the relational data model and temporal extension of SQL.

In fact, major database packages today include temporal support. Similarly, there are extensive research efforts underway for incorporating temporality into the Semantic Web data model, namely RDF and its variants. However, this is a challenging issue since RDF data model is hard-wired as triples. Handling temporality in RDF requires reification although semantically sound reification has a high overhead.

Semantic Web is formulated as layers. RDF is the fundamental knowledge representation model at the base. RDFS and OWL augment this base to provide more representational power, such as being able to specify class and property hierarchy, and more object property restrictions. Higher levels of abstraction provide even more representational capabilities. In this survey, we review temporal extensions proposed for RDF-based data models. For each model, we focus on core model components, syntax, semantics and its query language. *Valid Time* is commonly considered in proposed temporal extensions of RDF. A *Valid Time* timestamp augments a RDF triple, and it represents the time period for which the triple is valid. In the remainder of the paper, we focus on *Valid Time* since *Transaction Time* is not considered in temporal models we review. Nevertheless, applications in Semantic Web would benefit from RDF datastore augmented by the transaction time as well. For instance, when a RDF datastore is being constructed or RDF data collected progressively along the time dimension, a sequence of datastore states is indexed by the corresponding transaction time. The history of datastore changes can then be built and examined when needed. As a result, the database can be returned to any previous state

by the *roll back* operations. Due to the distributed nature of the Linked Data environment, the transaction time becomes even more important in tracking the changes in component datastores. The network latency introduces challenges in reconciling these changes. The benefit of transaction time is obvious. However, adding it to RDF is extremely complicated, both conceptually as well as implementing it. That is why all of the proposed extensions do not consider the transaction time and focus only on the valid time.

Our contributions in this paper include:

- Our survey is an up-to-date and comprehensive coverage of temporal Semantic Web models. There are survey papers of temporal Semantic Web models, such as [14] and experimental evaluations of [49]. However, new research has been reported since.
- We adopt a comparative framework in evaluating proposed temporal models.
- We have developed a taxonomy for classifying RDF temporal models. This taxonomy is based on the concept of reification which manifests itself as *Explicit Reification* and *Implicit Reification*. In the *Implicit Reification* case, we have identified three subgroups: (1) Instantiating-Identifying Concept/Relationship,(2) Relationship Entity Conversion, and (3) Named Graphs.

2 RDF Basics

Resource Description Framework (RDF) [2] is a graph-based data model. Its basic construct is a simple triple that is made up of (subject, predicate, object) which makes an assertional statement. A collection of triples constitutes a RDF graph. In a RDF graph, the subjects and objects are visualized as vertices and the predicates as edges. These element may be resources identified by International Resource Identifier (IRI) [13], *blank nodes* denoted by locally scoped identifiers, or typed literals.

RDF model is formed by layered sets of vocabularies that define the specific meanings to an ontology and an entailment regime [27]. RDF vocabulary (rdfV) is a simple vocabulary that mainly includes a type predicate (rdf:type) and a property class (rdf:Property). RDF

Schema (RDFS) extends rdfV and introduces classes for collections of related resources, their relationships and property specifications. Web Ontology Language (OWL) [3] further extends RDFS and provides more expressiveness. OWL-lite, OWL-DL and OWL-Full are the three versions of OWL family [56] with increasing expressive power and computational complexity.

The formal semantics of the Semantic Web layered cake is defined accordingly for each semantic extension. Model-theoretic semantics is used to define an interpretation model for each extension. In addition, an interpretation model is used to characterize an entailment regime. An entailment, also known as logical consequence, is an implicit relationship that can logically be inferred from the the given statements in a RDF store. Given two RDF graphs G_1 and G_2, if every RDF interpretation satisfying G_1 also satisfies G_2, the graph G_1 entails the graph G_2, denoted by $G_1 \models G_2$ [27]. For rdfV, two entailment rules, rdfD1 and rdfD2, are defined [27]. For instance, based on rdfD2 entailment rule, the running example (:John, :enrolled, :SW) entails (:enrolled, rdf:type, rdf:Property). Furthermore, thirteen RDFS entailment patterns are defined for RDFS [27].

SPARQL Protocol and RDF Query Language (SPARQL) [46] and its newer version SPARQL 1.1 [24] are the main query language for RDF and RDFS. SPARQL has a similar syntax form to SQL, but it is specifically tailored for graphs. SPARQL provides graph pattern specifications and a SELECT construct to retrieve matched graph segments from a RDF ontology.

A rule language, Semantic Web Rule Language (SWRL) [31], was also developed. SWRL combines OWL-DL and Rule Markup Language (RuleML). It expresses rules in OWL. The rule form *antecedent* $\Longrightarrow$ *consequent* expresses that the consequent holds if the antecedent is true. As an example, a new predicate *hasGrandParent* can be defined by a rule:

$$\mathit{hasParent}(x1, x2) \wedge \mathit{hasParent}(x2, x3) \Longrightarrow \mathit{hasGrandParent}(x1, x3)$$

This rule simply states that if an individual, $x1$, has a parent who has a parent, $x1$ has a grandparent.

2.1 Reification

The verb *reify* originates from *res* in Latin, meaning to *thingify* or to convert into a concrete thing. It is commonly used in different disciplines. For instance, in First Order Logic, *reification* generally refers to the use of terms to express concepts that are normally represented using predicates [17]. In other words, it allows making an assertion about a predicate. RDF model is confined to binary predicates, hence reification is needed to represent higher arity predicates. A RDF triple (a binary predicate) makes an assertion about a subject and an object. We can consider it as an atomic statement. When we want to make another assertion about an atomic statement, we need a new construct in RDF. That is reification. In general, the reification process starts by making a given statement bind to a new identifiable *resource* (i.e., an identifier is used to represent the statement) which acts as a proxy for the statement. The proxy then can further be used to assert properties on behalf of the statement.

Consider the facts given in Table 1: John enrolled in the Semantic Web class when he lived in NYC from 2/1/2016. Clearly, this statement asserts several facts and can not be expressed in one RDF triple. For instance, a binary predicate *enrolled* (John, SW) represents part of this fact. If the predicate *enrolled* is reified, it becomes a new term that can be used consequently as a component in other assertions. We explain the two forms of reification by an example.

There are two types of reification in RDF: implicit reification and explicit reification. We illustrate both forms of reification by using the facts given in

Person relation of Table 1. Table 1 represents a person whose name is John. For simplicity, we also use the term *John* as an identifier. This individual enrolled in SW, lived in NYC, and had a validity interval 2/1/2016–5/31/2016. *Person*(John, SW, NYC, 2/1/2016–5/31/2016) is a 4-ary predicate that represents the relation given in Person table. Obviously, RDF can not represent it directly, so it needs to be broken

Table 1 Person relation

Name	Enrolled	LivedIn	HasDate
John	SW	NYC	2/1/2016–5/31/2016

into several triples by using reification. Implicit reification allows defining binary predicates shown in Figure 1. Obviously, implicit reification breaks any n-ary relationships into several binary relationships. This is similar to representing a n-ary relationship in Entity Relationship Data models into corresponding binary relationships.

All of these predicates can directly be represented in RDF. *enrolled* and *livedIN* predicates are clear; however the last predicate *hasDate*(John, 2/1/2016–5/31/2016) asserts that John has a date 2/1/2016–5/31/2016. It is not clear whether it is for *enrolled* or *livedIn*, or both, or something else. Resolving this ambiguity requires explicit reification that is also provided in RDF. Considering that the date value 2/1/2016–5/31/2016 actually applies to John's enrollment in SW, this fact is reified as given in Figure 2.

In explicit RDF reification process, a triple is instantiated as a new resource that belongs to the class *rdf:Statement*. According to RDF specification, the new resource required in reification can be written as a blank node or identified by an IRI. In the latter case, such an IRI does not represent any concrete realization of a triple or resource. The original triple can then be associated with additional properties as if it is a standard resource. Figure 2 depicts the reification of a simple triple (John, enrolled, SW). First a new identifier :stmt1 is defined. This identifier represents the triple, (:John, :enrolled, :SW), which is further augmented by meta properties. Hence, components of the original triple become objects of special meta properties, including *rdf:subject, rdf:predicate and rdf:object*. The new resource *:stmt1* can be described by additional properties, such as the occurring time (2/1/2016 to 5/31/2016), place, certainty or provenance.

According to RDF specification, '*the reification of a triple does not entail the triple, and is not entailed by it*' [27]. However, from the reification in Figure 2, there is an entailment pattern to infer the original triple of (John, enrolled, SW). Since RDF specification

```
enrolled(John, SW)
livedIn(John, NYC)
hasDate(John, 2/1/2016-5/31/2016)
```

Figure 1 Binary predicates for person relation.

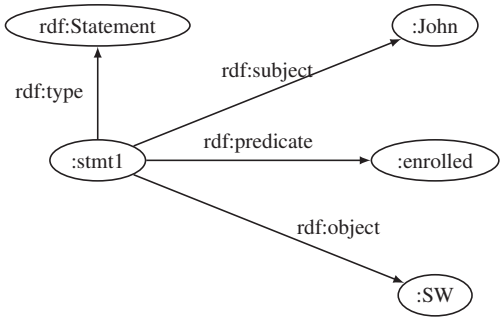

Figure 2 RDF reification.

does not constrain the semantics of standard reification, it is the user's decision to accept the entailment pattern or not.

Moreover, reification also suffers from the proliferation of extra objects and triples that are needed for representing higher order relations. That is, to reify (John, enrolled, SW), four additional triples are needed before additional facts can be added. As a result, the graph size increases. And even worse, the reified graph makes queries more difficult to write as we will see it in an example later.

For the user's convenience, two presentations of RDF graphs are commonly used in the literature. Figure 3(a) includes the original triple instead of converting the predicate *:enrolled* to an object. In contrast, Figure 3(b) uses a node connecting to an edge instead of another node. This treatment is a violation of the general definition of graphs, and common graph-based operations can not be applied directly.

3 Time Basics

Time is a very pervasive concept. It is naturally continuous. However, for the sake of representation, Time is usually modeled as a discrete sequence of time instants. Continuous time instants are combined into intervals for a compact representation. Among many approaches to model temporal data and knowledge, one can choose to incorporate *Time* into a model as the first order citizen, i.e., *Time* is an explicit part of the model and its language. Alternatively, *Time* can also be realized implicitly by capturing temporal ordering of different model

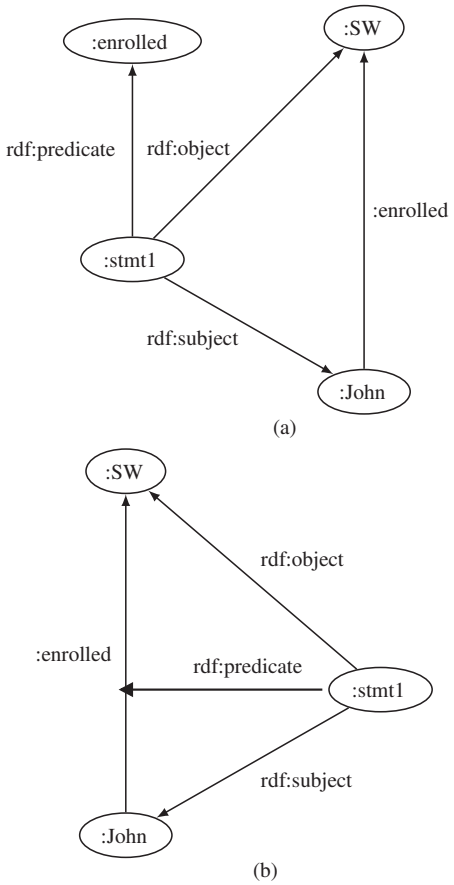

Figure 3 Graph representation of RDF reification.

states. That is, when the model changes, an updated version of the model is generated, while the original model is preserved. This leads to a notion of *versioning*. However, versioning suffers from the rapid proliferation of state objects which are prohibitive when considering the large size of RDF stores. That is why versioning is not generally used in Semantic Web to track time varying data and knowledge. One notable exception is Named Graph [11], which comes close to versioning. As a result, most of the temporal models for the Semantic Web employ more explicit ways of incorporating time into the model,

instead of versioning. Nevertheless, versioning can be used in tracking the changes of the ontology, not its data as illustrated in the reference [19, 57].

We assume that the time domain is linearly ordered in a single time dimension. Considering this structure: $T_P = \{T, <\}$, where *T* is the set of time instants and $<$ is the linear order on T. T_P refers to a domain of time points. For time representation in this paper, we use time instants with the granularity of one day as metrics for T_P. All the time intervals, denoted by T_I, are defined as:

- A Time interval contains a set of consecutive time instants between its boundaries. A time interval $[t^-, t^+]$ is the set $\{t_k | t_k \in T_P, t^- \leq t_k \leq t^+\}$ and it is closed at both ends. A time interval may be open on either end or both: $[t^-, t^+)$, $(t^-, t^+]$ or (t^-, t^+).
- t^- is the beginning instant which sets the minimal boundary of the time interval. t^+ is the ending instant which sets the maximal boundary of the time interval.

4 Time in Semantic Web

In this section, we review Semantic Web temporal models reported in the literature. We examine the characteristics of each temporal model and develop a taxonomy to categorize them into two groups: explicit reification-based and implicit reification-based. Explicit reification-based temporal models employ standard RDF reification with either RDF reification vocabulary, such as rdf:subject, rdf:predicate, and rdf:object, or special functions to transform the proposed temporal model to an equivalent standard RDF model. In comparison, temporal models in *implicit reification* group employ some mechanism to identify a concept, a triple, a relationship or a graph. As we shall see later in this section, different logical constructs are employed to conceptualize an RDF graph at different levels, such as triple, and graph level. This group is further characterized into three subgroups by considering the underlying semantics they use: (1) Instantiating-Identifying Concept/Relationship, (2) Relationship Entity Conversion, and (3) Named Graphs.

For each model, we consider core model components, extensions to RDF/RDFS vocabularies, SPARQL query support and special features, if any. As we will see latter, some proposals employ W3C recommended OWL-Time ontology [12] as the time domain ontology whereas others include their own definition of time.

4.1 Running Example and Query

The data in Table 1 will be used for illustrating each temporal model. A simple RDF triple, (John, enrolled, SW), asserts the fact about an individual *Student* John and his enrollment in a Semantic Web class. The predicate *enrolled* relates a *Student* instance to a *Class* instance (i.e., as domain and range respectively). Furthermore, a closed interval [2/1/2016, 5/31/2016] needs to be associated with this triple. John may enroll in other classes over time. Hence, it is conceivable to have another triple: (John, enrolled, OOP) making another assertion about his enrollment in other time, such as [8/31/2016, 12/22/2016]. We use standard U.S. time representation for the timestamps in the examples. For each proposal, we will use standard RDF to depict the running example. A bubble denotes a resource as a subject or as an object. Directed lines represent a predicate. Basic data values are represented by rectangles.

The query "*retrieve the valid time when John enrolled in the Semantic Web class*" will also be used as a running example to illustrate how it is expressed in each temporal model and its version of SPARQL.

Namespace and Prefixes The following namespaces, IRI declaration and prefix taken from [1, 13] will be used as a common notation in all of the examples:

```
rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
rdfs: <http://www.w3.org/2000/01/rdf-schema#>
owl:  <http://www.w3.org/2002/07/owl#>
 :       <http://example.org/TemporalSW#>
```

The running example assumes the base ontology Namespace– http://example.org/TemporalSW, and use ":" as its prefix.

4.2 Explicit Reification Based Temporal Models

One of the early and formal extensions of RDF to handle temporality is Temporal RDF [21]. Later enhancements are introduced to this extension, such as [23, 32, 47]. In the following, we review Temporal RDF and its enhanced versions.

4.2.1 Temporal RDF

In Temporal RDF, each triple is timestamped with a time instant or an interval [21]. Timestamping is achieved by using standard RDF definitions and an internal time domain that includes temporal property specifications, such as *temporal, instant, interval, initial* and *final*. A Temporal RDF triple is in the form of $(s, p, o)[T]$ and visualized as a temporal RDF graph given in Figure 4. ":stmt1" and ":temporal_1" are ground nodes that substitute blank nodes, which are used as subjects in the original work for reification. Ground nodes refer to non-blank nodes.

In Figure 4, the triple (John, enrolled, SW) is reified by *:stmt1* of rdf:Statement class. *:stmt1*, is further associated with a temporal entity :temporal_1 and then an interval :i1. :i1 is the valid time interval of the triple which has its begin and end time instants "2/1/2016" and "5/31/2016" respectively, whereas natural numbers are used as time instants in the original work. This temporal fact is therefore represented by seven RDF triples in the case of time interval and by six triples in the case of time instant.

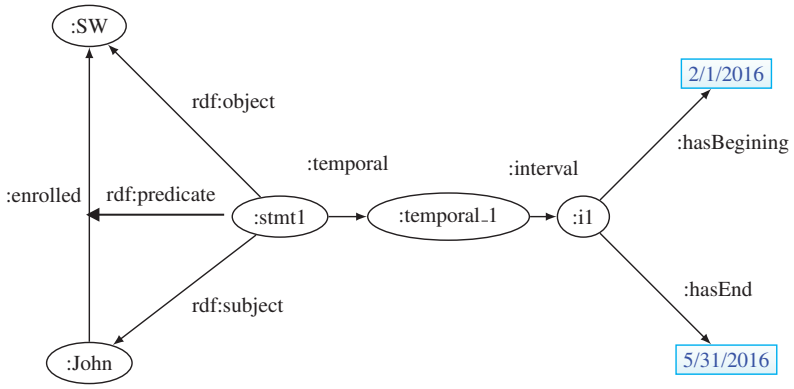

Figure 4 Temporal RDF.

The semantics of a Temporal RDF graph [21] is provided in terms of nontemporal RDF and RDFS graphs. Temporal entailment is defined based on the closure of temporal and non-temporal graphs. Specifically, a temporal graph G_1 entails G_2 if and only temporal closure of G_1 entails G_2. Furthermore, a deductive inference rule system for Temporal RDF graphs is outlined. Temporal rules are defined to equate an interval and a instant version of temporal graphs.

A query language proposed for Temporal RDF graphs is provided in a rule form. The running example query can be expressed conceptually as follows:

```
(:X, :interval, ?Y), (?Y, :hasBegining, ?ti),
(?Y, :hasEnd, ?tf)
<-- (:John, :enrolled, :SW):[?ti, ?tf].
```

Rewriting the above running example in SPARQL results in the following:

```
SELECT ?Y  ?ti ?tf
WHERE  { :John :enrolled  :SW.
?X    rdf:type   rdf:Statement.
?X    rdf:subject   :John.
?X    rdf:predicate  :enrolled.
?X    rdf:object  :SW.
?X    :interval  ?Y.
?Y    :hasBegining   ?ti.
?Y    :hasEnd   ?tf.
}
```

In this query, presence of the original triple (John, enrolled, SW) is assumed. Nevertheless, the query result preserves it even if this assumption is dropped. Query processing and semantics are also defined as a temporal tableau similar to a language presented in [22]. The complexity of query processing is briefly explained. Moreover, the authors conclude that adding time to the proposed Temporal RDF does not dominate the computational complexity.

4.2.2 Enhanced Temporal RDF

The Temporal RDF model [21] is enhanced by allowing anonymous timestamps [23] for temporal triples. The role of anonymous timestamp is similar to that of the *blank node* in RDF model. That is, a temporal triple has a time but the exact boundaries of this time are not specified. Such a temporal triple is represented in the form of (s, p, o):[X] which asserts that (s, p, o) is valid during some unkown time [X]. *General temporal graphs* are similarly defined as temporal graphs with known or anonymous timestamps. The *t-ground* general temporal graph is defined as one that does not contain anonymous timestamps.

The semantics of general temporal graphs is given similar to Temporal RDF semantics developed in [21] and it includes an additional slice closure of general temporal graphs. Slice closure of a general temporal graph is computed by a non-temporal closure of snapshot graphs for each time point. The complexity of evaluating entailment for general Temporal RDF graphs is *NP complete* [23]. Query language for the general Temporal RDF is similar to the example shown above for Temporal RDF.

4.2.3 C-Temporal Graph

Temporal RDF model [21] is further extended to include temporal constraints and reasoning [32]. A C-Temporal Graph is denoted by $C = (G, \Sigma)$. G is a temporal graph that contains temporal triples and Σ includes temporal constraints enforced on the triples of the graph G [32]. *Temporal blanks* are introduced as RDF nodes that contain anonymous time information represented by time variables. The treatment is similar to the Enhanced Temporal RDF that handles anonymous timestamps [23]. As an example, a student went to high school at an unknown time T_1, and later he went to college at another unknown time, T_2. These two facts are represented as two Temporal RDF triples with the variable timestamps T_1 and T_2 respectively. To preserve a proper temporal order, a constraint, $T_2 > T_1$, is enforced in the model. As in [21] the temporal entailment is based on the closure of temporal and nontemporal graphs, a C-Temporal graph can also be converted to a temporal closure defined in [23]. In such case, the temporal entailment can be handled as usual.

Additionally, query processing for the C-Temporal graph is reduced to matching query patterns on the closed graph.

4.2.4 tRDF for Indeterminate Triples

The tRDF model [47] is based on the Temporal RDF proposed earlier in [21, 23]. tRDF particularly supports another type of *anonymous timestamp* in indeterminate triples. A *determinate triple* (*s, p, o*)[*T*] represents that the triple is always *valid* in the interval T. In contrast, an *indeterminate triple* (*s, p* : [*n* : *T*]*, o*) represents that the triple is *valid* at most n distinct time points in the interval T. A tRDF graph includes both determinate and indeterminate triples. In addition, the concept of normalizing a tRDF graph is defined in order to preserve good properties of the tRDF model. Normalizing tRDF employs the notion of *value-equivalent-tuples* from temporal databases [33]. Two tRDF triples are value-equivalent if their non-temporal parts are identical. As an example, suppose John actually enrolled in Semantic Web class twice: (*John, enrolled, SW*)[T_1] and (*John, enrolled, SW*)[T_2]. Instead of storing them as two value-equivalent triples, *coalescence* is applied to merge overlapping or connecting intervals into a single cumulative interval, i.e., $T_1 \cup T_2$. The consolidated interval can then be used as the timestamp of a single representative triple (John, enrolled, SW)$\{T_1 \cup T_2\}$. As a result, a normalized tRDF graph G entails each one of the coalesced tRDF graphs.

Indexing structure, tGRIN, is proposed to improve the performance of tRDF triple storage in the query evaluation. A tGRIN index is a balanced tree structure that stores *close* graph vertices together in the same index node. The closeness of two resources x and y is determined by a distance metric that combines general graph distance, $d_G(x, y)$, and temporal graph distance, $d_T(x, y)$, by a *k-norm* function, $[d_G(x, y)^k + d_T(x, y)^k]^{1/k}$ [47]. Experiments show that tGRIN index structure for tRDF queries outperforms the standard B-tree index structure used in traditional relational databases [47].

The formal semantics and querying tRDF are based on equivalent models developed in Temporal RDF [21, 23]. Additional semantic conditions are also needed to interpret indeterminate triples and queries.

4.2.5 Generalized RDF Annotation

In [59], a generalized framework for representing and reasoning annotated RDFS is proposed. This model is based on the works of annotated RDF (aRDF) [55] which employs *annotated logic* [34], and its query language, AnQL [36]. The framework represents an abstract form of triples: (*s, p, o*)[*L*] where *L* is an annotation term that belongs to a domain *D*. The annotation domain *D* could be temporal, fuzzy, combinations or others. It is defined as an algebraic structure: $D = \{L, \preceq, \bigwedge, \vee, \otimes \Longrightarrow, \bot, \top\}$where elements in L are annotation terms: L = [0,1] and L = [T] for fuzzy and temporal domains respectively. The top $\top$, bottom $\bot$, order $\preceq$, meet operator $\bigwedge$, join operator $\vee$ and t-norm $\otimes$ are used for constructing the annotation domain and its inference patterns. The t-norm $\otimes$ is used for combining annotation information. For instance, based on RDFS entailment pattern-rdfs5 of [27], (a, rdfs:subPropertyOf, c):$L_1 \otimes L_2$ can be inferred from (a, rdfs:subPropertyOf, b):L_1 and (b, rdfs:subPropertyOf, c):L_2. The annotation term in the inferred triple takes the conjunction of L_1 and L_2. For the temporal case, $\otimes$ is overloaded to represent the intersection of time intervals. Multiple annotation domains may be combined using the generalized framework. A complex domain D can be constructed from individual annotation domains: $D = D_1 \times D_2 \times .... \times D_n = \{L, \preceq, \otimes, \bot, \top\}$. The model is also augmented by a set of inference rules for annotated RDFS.

4.2.6 RDF*

Hartig proposed extensions of the RDF model and SPARQL to represent statement-level metadata [25]. The RDF* model allows nested triples. That is, a triple can be embedded as a subject or an object in another triple. Figure 5 depicts the running example. The original triple, (John, enrolled, SW), is nested in an abstract object. To accommodate its nested structure, RDF* requires syntactic and semantic extensions of RDF model. Nevertheless, RDF* graphs can be transformed to standard RDF graphs by a set of special functions provided in [25]. They are blank node assignment function, reification function and unfold function. A transformed RDF* graph is an explicitly reified standard RDF graph and would be similar to the example in Figure 2. In addition, Turtle* and SPARQL* are proposed as extensions of standard Turtle and SPARQL

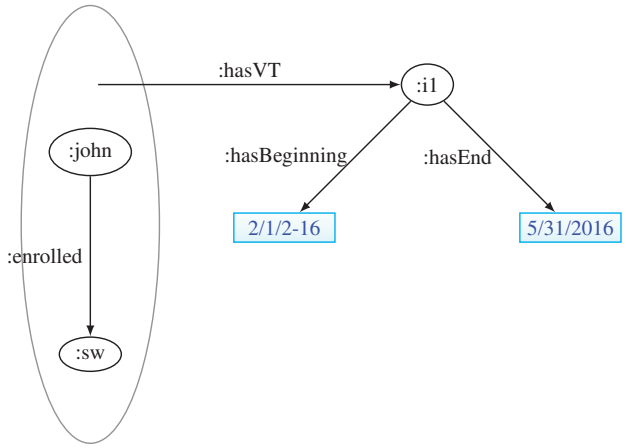

Figure 5 RDF*.

notations respectively. The running example query can be written in SPARQL* as follows. Double bracket pairs show a nested triple:

```
SELECT ?ti ?tf
WHERE  { <<:John :enrolled  :SW>> :hasVT ?i1.
?i1  :hasBeginning   ?ti.
?i1  :hasEnd         ?tf.
}
```

4.2.7 YAGO 2

The original YAGO Knowledge Base [52] is constructed automatically from articles on Wikipedia. Each simple article on Wikipedia belongs to a article category, and mainly contains a lead section, a content body, appendices and bottom notes [4]. An article becomes an entity in YAGO. Article categories on Wikipedia provide the *type information* for it. The type information is linked to the taxonomy of WordNet [30]. In YAGO, each fact is represented by a triple, (S, P, O). Each fact is also reified, so an triple identifier is assigned. This effectively results a quadruple: (id, S, P, O).

YAGO 2 is the new version of YAGO. YAGO 2 employs an extensible extraction architecture that is based on declarative rules, whereas YAGO's extraction rules are hard-wired to the source code

[30]. In addition, YAGO 2 incorporate both *temporal* and *spatial* dimensions to the knowledge base. For the temporal dimension, *yagoDate* is the main data type that denotes time points in days. A time interval can be represented by two time points with a pair of relations, such as *OccursSince* and *OccursUntil* [30]. *Entity time* denotes the existence in time of an entity while *Fact time* represents the valid time of a fact. Only the *fact time* is applicable to the running example. The running example can be represented similarly to Temporal RDF of Figure 4.

4.3 Implicit Reification Based Temporal Models

In general, implicit reification based models use different types of abstraction for handling reification. They do not employ reification vocabularies of RDF specifications. Two subcategories follow.

4.3.1 Instantiating-Identifying Concept/Relationship (IIR)

In IIR models, a concept or relationship is reified and further temporalized. Either such relationship is abstracted as a new object, or a concept is viewed as four dimensional and instantiated to have temporal extents. *Singleton Property* converts each relationship to be universally unique. *4D fluents* use concepts that view each resource as a perdurant. Fluents represent properties that change over time.

4.3.1.1 *Singleton Property*

Nguyen et al. propose the concept of singleton property for representing and querying meta knowledge in [41]. This approach recognizes each RDF triple as an unique relationship and introduces multiple contextual instances to it as needed. Given a relationship between two objects under a context, a singleton property is introduced to denote the relationship instance with a specific context, such as temporality, provenance, etc. In other words, a singleton property is an instance of a given relationship used to assert the context property values. Figure 6 shows the running example in singleton property approach.

The unique property :enrolled_1 in Figure 6 is an instantiation of the generic *:enrolled* by *rdf:singletonPropertyOf*. It is then used for

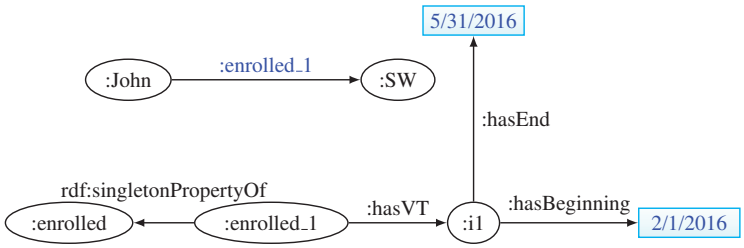

Figure 6 Singleton property.

asserting the relationship between John and the class SW. The temporal context is therefore asserted by (:enrolled_1, :hasVT, :i1).

The formal semantics of the singleton property is derived from the standard RDF and RDFS semantics with the additional semantics extension for the vocabulary *rdf:singletonPropertyOf*. The singleton property gives rises to three cases of query patterns: data, metadata and mixed patterns which SPARQL supports. The running example query belongs to the metadata pattern and can be written in SPARQL as follows:

```
SELECT ?ti ?tf
WHERE  { ?p  rdf:singletonPropertyOf :enrolled.
:john ?p  :sw.
?p  :hasVT ?i1.
?i1  :hasBeginning ?ti.
?i1  :hasEnd ?tf.
}
```

4.3.1.2 *4D Fluents*

Welty et al. proposed 4D Fluents model for representing time-varying relationships in OWL [58]. This model employs *4D view* and *Fluent*. Haynes introduced *four dimensional view* or *perdurantist view* into Computer Science in his seminal work [28]. *Perdurantism* is a philosophical theory of persistence and identity [26], and it is closely related to *four dimensionalism*. In *four dimensional view*, an object that persists through time has distinct temporal parts at every time instant through its existence in time. Furthermore, each persisting object can be considered a *four dimensional spacetime worm* that stretches across space-time.

Slicing the worm at a specific time interval or instant of the time dimension yields a temporal part. A Temporal part is also called a *time slice* in other literature [16]. The slicing produces *entity-at-a-time*. In contrast, *three dimensional view* considers that an object is wholly present or endures through its existence in time. Therefore there are no temporal parts.

Fluent is a component of *Situational Calculus* which is a logical language for representing change. McCarthy first introduced *Situational* Calculus [38, 39]. It concerns situations, actions and fluents in a dynamic domain. *Actions* make the domain change from one situation to another. Fluents are situation-dependent functions for describing the effects of actions.

In 4D Fluents model, *fluents* are properties that change over time [58]. These properties are special cases in that both the domain and range of them are temporal parts of the corresponding entities. *TemporalPart* is the main class for converting regular entities to 4D spacetime worm ones. OWL-Time ontology of [29] is used as time domain in 4D Fluents model. Particularly, a class *TimeInterval* derived from the equivalent class of OWL-Time is used for all temporal terms.

Several object and fluents properties are listed in Table 2. Figure 7 represents running example in 4D Fluents model.

In Figure 7, individuals :John and :SW are two *4D entities*. Each entity has temporal parts, :John@i1 and :SW@i1 respectively. The property *:enrolled* is transformed to a fluent whose domain and range are both *temporal parts*. Each temporal part is associated with a specific temporal extent, i.e., time interval, that denotes its valid time. One fluent property requires two extra objects, i.e., temporal parts, and two properties, in contrast to reification, that uses one extra object and four properties as illustrated in Figure 2. Moreover, the 4D fluents model has advantages. Particularly, OWL inverse operator and cardinality

Table 2 4D fluents ontology object property

Object Property	Domain	Range
:fluentProperty	:TemporalPart	:TemporalPart
:temporalExtent	:TemporalPart	:TimeInterval
:temporalPartOf	:TemporalPart	complementOf(:TimeInterval)

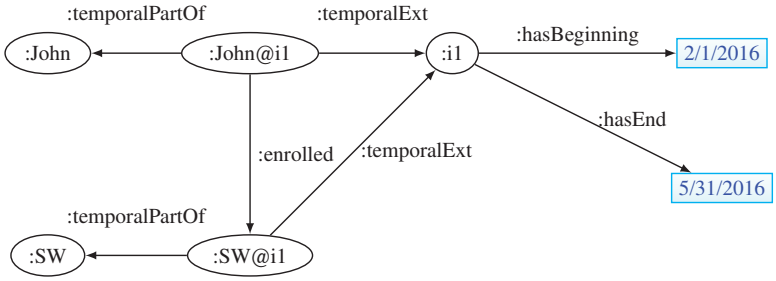

Figure 7 4D fluents.

constraints are available and standard OWL reasoners can be used for inferencing.

The 4D Fluents model is within standard RDF and OWL-DL. Its semantics is defined based on OWL-DL semantics. Consequently, there is no need to extend RDF or OWL. The running example query can be written in SPARQL 4D Fluents model:

```
Select ?ti  ?tf
WHERE  {?ts1  :temporalPartOf :John.
?ts2   :temporalPartOf  :SW.
?ts1   :enrolled  ?ts2.
?ts1   :temporalExt  ?i.
?ts2   :temporalExt  ?i.
?i     :hasBeginning ?ti.
?i     :hasEnd ?tf.
}
```

Since the 4D Fluents model imports OWL-Time [29], :i1 in Figure 7 is an OWL-Time interval, while ?ti and ?tf in the above query are two OWL-Time instants.

4.3.1.3 *Extended 4D Fluents*

Batsakis et al. extended 4D Fluents model to incorporate qualitative temporal relations that have unknown temporal information [7, 8]. Such a relation is considered an object property between time intervals. The model employs OWL-Time [29] and Allen's thirteen temporal relations [5], such as before, meets and overlaps, etc. Consider the

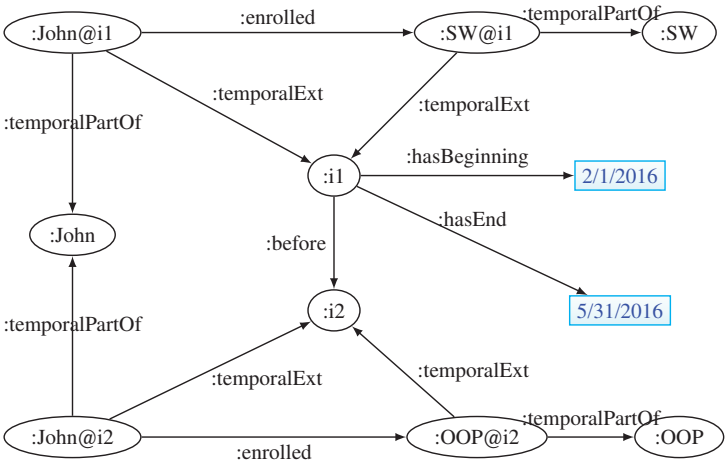

Figure 8 Extended 4D fluents.

running example and additionally the triple that John enrolled in another class OOP in a later semester. However, the actual enrollment time was unknown. In Figure 8, time interval *:i2* denotes the valid time of John's OOP enrollment and its relationship to *:i1* is captured by the object property *before*. Semantics of the extended 4D Fluents model is based on the original 4D Fluents model, with the additional temporal semantics needed for qualitative temporal relations.

TOQL [6] is the SQL-like query language for Extended 4D Fluents model. To accommodate querying qualitative temporal relations, additionally query constructs, such as "AT" clause and Allen temporal operators [5], such as before, after, meets, etc., are included in TOQL.

4.3.1.4 *Temporal Web Ontology Language-tOWL*

Fransincar et al. proposed an extension of the OWL-DL language, called tOWL [40], for representing time and changes in an ontology. tOWL uses a subset of OWL-DL whose foundation is the logic $\mathcal{SHIN}$ (D). This logic is sufficiently expressive and is decidable for a sound and complete reasoning algorithm [37]. The time domain of tOWL handles both instants and intervals that are modeled by rational numbers and a set of partial order relations over them. As a result, an actual time instant,

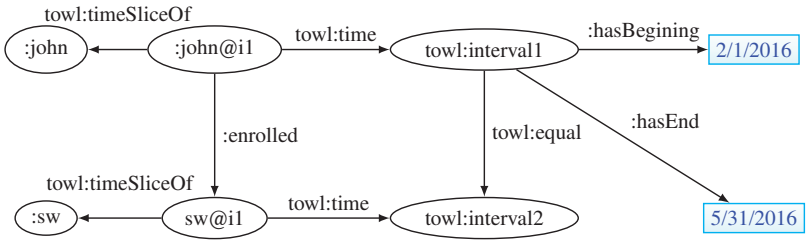

Figure 9 tOWL.

an interval or Allen's temporal relations [5] are all converted to rational number-based equivalent instants or relations. For modeling changing values, tOWL employs the 4D Fluents model, i.e., perdurantist's view [58]. tOWL is conceptualized as a layered approach. The foundation layer is OWL-DL and the extended concrete domain is the second layer. Time representation is at the third layer, which is defined by the concrete domain.

Figure 9 gives the running example in tOWL. Note that tOWL requires separate intervals for :John and :SW. Therefore, a restriction on the equivalence of towl:interval1 and towl:interval2 is enforced by a relation *towl:equal* in Figure 9 which is one more triple used compared to 4D Fluents model in [58].

In tOWL, the time domain is based on rational numbers and relations over them. This approach makes tOWL more expressive in representing complex temporal relations. For instance, in Figure 9, a temporal constraint *towl:interval1 equates towl:interval2* can be expressed with the equality of endpoints of the two intervals. Additionally, tOWL reduces the proliferation of objects by differentiating types of fluents as FluentObjectProperty and FluentDatatypeProperty. For a FluentDatatypeProperty, which relates a time slice to a typed value, three triples can be saved due to that the time slice is not needed for a typed value.

4.3.2 Relationship to Entity Conversion (REC)

In REC models, a relationship is transformed to a *composite entity*. The transformed entity comes in two forms: a new entity that implicitly reifies the original triple, or an abstract object that becomes a term for

further use. As an example of REC models, N-ary relations provide a main modeling concept: a triple is *objectified* as a new entity and can further be associated to properties, such as time.

4.3.2.1 *N-ary Relations*

In principle, N-ary relation is a generalization of reification. For each N-ary relation, a new class with an instance is introduced for it as if the relation is objectified. Further property assertions can be made with respect to the newly introduced instance. Figure 10 gives the running example in N-ary relations.

The resource *:enrolled1* in Figure 10 is introduced as a new instance encapsulating both the course name value, SW, and its valid time interval through two properties, *:hasCourse* and *:hasVT*. The relation (:John, :enrolled, :SW) is converted to an entity class *:Enrollment*. The property *:enrolled* is overloaded, so its range becomes the newly introduced class *:Enrollment*. Adding time to the original triple, i.e., (John, enrolled, SW), requires three more triples.

N-ary relation approach does not require extension to RDF, RDFS or OWL vocabularies. It simply converts relationships to entities that encapsulate properties. The semantics for N-ary relation approach is based on RDF and RDFS semantics. In Figure 10, the new object :enrolled1 may also be represented by a blank node. A blank node does not have any meaning, but acts like a wrapper for grouping related objects.

The running example query in SPARQL is as follows:

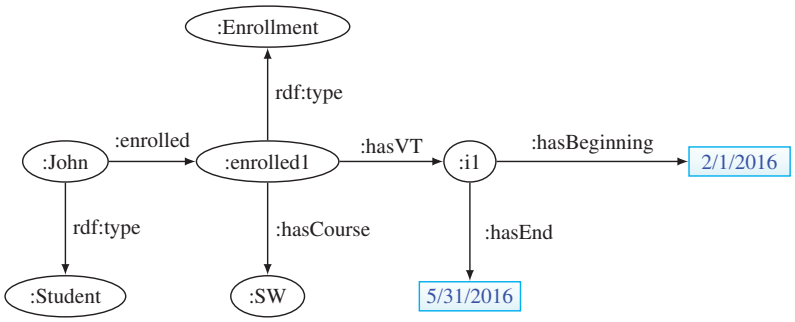

Figure 10 N-ary relation.

```
Select ?ti  ?tf
WHERE  {
:John    :enrolled    ?e.
?e    rdf:type  :Enrollment.
?e    :hasCourse    :SW.
?e    :hasVT   ?i.
?i  :hasBeginning ?ti.
?i  :hasFinish ?tf.
}
```

While N-ary relation approach can be applied to OWL, it would incur overheads. For instance, multiple inverse properties are needed for a N-ary relation. Moreover, the use of cardinality restrictions becomes limiting on some roles that depend on the class of some other roles [42].

4.3.2.2 *FrameBase*

FrameBase [48] integrates FrameNet [15] and WordNet [35] for constructing an extensible RDFS schema. FrameNet originated from *Frame Semantics* [15]. Frame Semantics assumes that people understand the meaning of words by evoking semantic frames, and relate words to meanings. FrameNet is a large lexical database that contains semantic frames for describing meanings of natural language words. It also provides example sentences annotated with frames and frame elements to demonstrate the use of words in the frame. Each distinct semantic frame contains *lexical units* and *frame elements*. Lexical units are the *keywords* used to evoke the frame, while frame elements are the roles that describe properties of the frame.

WordNet [35] is another well known lexical database for English that provides meanings of words. Each type of words, such as nouns, verbs, adjectives etc. is organized to form a *synset*, or a synonym set. Each set represents a lexical concept [35]. WordNet contains about 117,000 synsets, and each may be linked to others. The major relation is *hyperonymy*,or *is A* relation, and *meronymy*, or *part-whole*. These relations, among other components, form a lexical network of words and concepts.

The main representation model used in FrameBase is similar to the model of N-ary relations discussed above and in Figure 10. A primary

entity is created to form a semantic frame for the N-ary relation. The frame's properties can be asserted in a sense of *semantic role* [18, 42]. A mapping between FrameNet and WordNet is created to form the basis lexical units and relations for FrameBase's schema.The mapping is further transformed by the *schema induction* and *automatic reification-dereification mechanism* to yield a light weight yet broad covering frames [48].

4.3.2.3 *Valid-Time Temporal Model*

O'Connor et al. propose a valid-time temporal model and a SWRL-based [31] query mechanism for manipulating temporal knowledge in OWL ontologies [44]. The model introduces a new class, *temporal-Fact*, and uses N-ary relations. The running example is represented in Figure 11. The graph in Figure 11 is similar to N-ary relations in Figure 10. However, there are differences on the class hierarchy. In the validtime temporal model, any existing OWL class can have temporal aspects as long as it subclasses *temporal:Fact*, which is the super class of all temporal facts. This avoids significant ontology rewriting in converting an ontology to a temporal version.

The temporal expressivity of this model is further enhanced by using SWRL [31] to construct temporal rules. A set of temporal operators that includes Allen's operators [5] is implemented as library-like *built-ins*

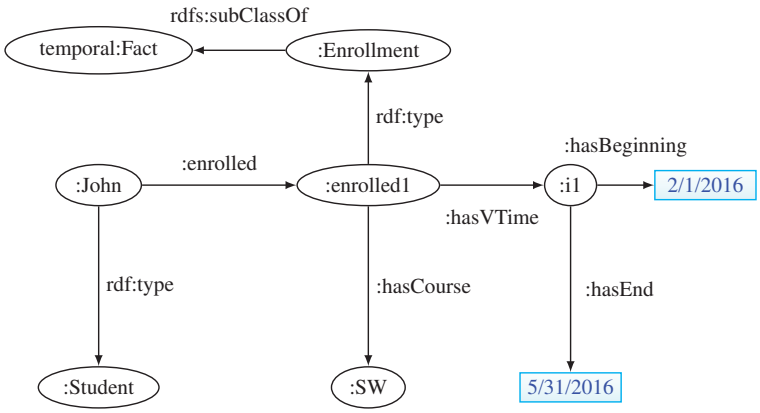

Figure 11 Valid-time OWL model.

for SWRL rules. With the temporal ontology and temporal built-in operators, complex temporal rules can be constructed.

Querying the temporal ontology is done by SQWRL [43]. All SWRL built-ins [31] are included for SQWRL, so complex temporal queries can be formed. These include queries that require complex closure, negation, or complex aggregation and grouping. The running example query is adapted to show temporal operator usefulness in this model. The following SQWRL query retrieves all resources who took the course :SW before 2016. The symbol ^ denotes logical *and*.

```
Student(?s) ^ :enrolled(?s, ?e)   ^
:hasCourse(?e, :sw) ^
temporal:hasValidTime(?e, ?vt)  ^
temporal.before(?vt,  "2016")
--> sqwrl:select(?s)
```

The above query can be transformed to a standard but lengthy SPARQL query. However, there are very limited temporal operator supports in SPARQL. If done so, the temporal order, such as *temporal:before*, may need to be fulfilled by using literal value comparisons. This model is designed to be implemented at the users' level. There is no formal extension to RDF, RDFS or OWL model and vocabularies. However, the additional semantics for Allen's temporal operator [5] and the built-in SWRL rules need to be added.

4.3.3 Named Graphs

The term Named graph was first introduced in [11]. Named Graphs extend RDF model to provide a mechanism for identifying grouped RDF triples. A named graph, denoted by (u_1, G_1) where G_1 is a standard RDF graph that is named by an IRI u_i [11]. W3C has adopted Named Graphs model in SPARQL query language [24, 46]. With the Named Graph model, the running example is represented in Figure 12.

In Figure 12, :graph1 indicates a graph for which additional properties can be asserted. W3C adopted a line-based N-Quads as concrete syntax for RDF 1.1 Datasets [10]. The format is a quad: <subject, predicate, object, GraphName>. For querying Named Graphs,

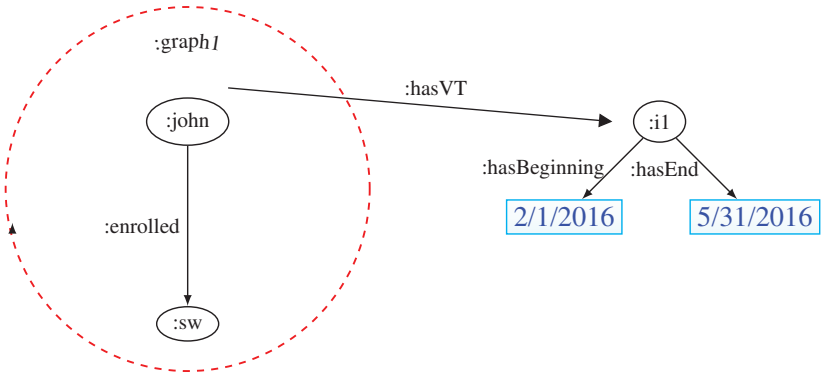

Figure 12 Named graph.

TriQL [9], RDFQ [51] and SPARQL are available. The running example query can be written in SPARQL as follows:

```
SELECT ?ti  ?tf
FROM  :graph1
WHERE  {:graph1    :hasVT  ?i.
?i1  :hasBeginning  ?ti.
?i1  :hasEnd ?tf.
:John  :enrolled    :SW.
}
```

Named Graphs model is a general purpose *triple grouping*. In the above query, it is assumed that all triples share the same temporal extent are grouped in the same graph. The *FROM* clause indicates the source graph. In an extreme case that each triple requires a different time reference, a significant amount of named graphs is needed. When triples may need multiple metadata annotation, using Named Graphs model becomes complex.

4.3.3.1 τSPARQL Temporal Queries

Tappolet et al. propose a temporal RDF query approach τSPARQL [53]. τSPARQL is defined as a shorthand format for querying such a temporal ontology. In this model, OWL-Time ontology [12] is used as the time domain which defines time instants and intervals. The target

temporal RDF model is designed using Named Graphs. Each graph is identified by exactly one time interval. Triples with the same temporal extent are grouped to the same graph. In other words, the *name* of a graph is the time interval. The grouping also introduces complexity to the model in that the indexing structure would have a significant impact on query retrieval time.

τSPARQL is based on SPARQL, and recognizes a *quadruple* form, such as ([ti, tf], s, p, o), as the main query pattern. The interval [ti, tf] is to be checked against *names* of graphs. The triple s, p, o are handled as a SPARQL query pattern. As a result, a τSPARQL query can be mapped to a standard SPARQL 1.1 one. The running example query can be written in τSPARQL as follows:

```
SELECT ?ti ?tf
WHERE {
[?ti, ?tf]  :John  :enrolled    :SW.
}
```

Assuming that the working example is represented as in Figure 12, mapping the above query to standard SPARQL results the following:

```
SELECT ?ti ?tf
WHERE  {
GRAPH  :graph1 {:John   :enrolled    :SW.}
:graph1   :hasBeginning   ?ti.
:graph1   :hasEnd   ?tf.
}
```

4.3.3.2 *RDF+*

RDF+ model [50] uses named graphs and triple-level identifiers. Named graphs are used in place of RDF reification. Triple identifiers allow explicit annotation of meta knowledge. RDF+ model has two type of statements: literal and meta knowledge statement. A RDF+ literal statement is a quintuple form (g, s, p, o, θ) where g is the graph's IRI, s,p,o are standard RDF triple components, and θ is a statement identifier. Based on the triple identifier in the RDF+ literal statement, the RDF+ meta knowledge statement can be formed as (θ, π, ω). θ is the literal

statement identifier, π is the meta knowledge property and ω is the range value of π. The set K of RDF+ literal statements and the set M of RDF+ meta knowledge statements constitute a RDF+ theory, (*K, M*) [50].

Bidirectional mappings between RDF and RDF+ are also defined in [50]. Our running example is mapped to the RDF+ model and results the following RDF+ literal statements and meta knowledge statement. Please note that we also adapt the N-triple-like syntax for representing a quintuple in the example. In addition, time intervals are assumed available although the original work uses time instants.

```
#  K <--RDF+   literal statements of
<:graph1> <:John> <:enrolled>  <:SW> <:stmtID1>.
<:graph2> <:graph1> <:timestamp>
<[2/1/2016,  5/31/2016]>    <:stmtID2>.

#  M<--RDF+  meta  knowledge statement
<:graph3> <:stmtID1>  <:timestamp>
"[2/1/2016, 5/31/2016]".
```

In the above mapping, :graph1 and stmtID1 both identify the original triple. :graph2 and :stmtID2 refer to the meta knowledge for :graph1. The statement in :graph2 is further stored as the associated meta knowledge in order to be compatible with standard RDF semantics. The formal semantics for RDF+ is provided by '*Meta Knowledge Interpretation and Model*' [50] that combines a standard interpretation I_s for statements in K, and a Π-interpretation for meta knowledge statements in M. An extension to SPARQL is also proposed. The running example query can be written as follows:

```
SELECT ?x
WITH META :graph3
FROM NAMED :graph1
FROM NAMED :graph2
WHERE  { GRAPH  ?g  { ?x    :enrolled   :SW }
```

The above extended SPARQL utilizes additional constructs: (1) an optional *WITH META* clause specifying the graphs which contains

the associated meta knowledge, and (2) the *FROM NAMED* clause that specifies the target graph for quadruple pattern matching [50]. The query evaluation system binds the variable ?x to the matched values based on the specified quadruple pattern. It further outputs values for all meta knowledge associated with the pattern, such as the following:

```
?x                timestamp
---------------------------------
:John     [2/1/2016,  5/31/2016]
```

5 Discussion

We have constructed a taxonomy depicted in Figure 13 for classifying the proposed temporal extensions to RDF and OWL. Table 3 is a concise summary of various characteristics. For summarizing and comparing them, we use the following characteristics: (1) RDF, RDFS or OWL extension of these proposals, (2) additional objects required, (3) number of triples needed, (4) formal semantics specified or not, (5) Time Domain, (6) Instant or Interval used, and (7) query language.

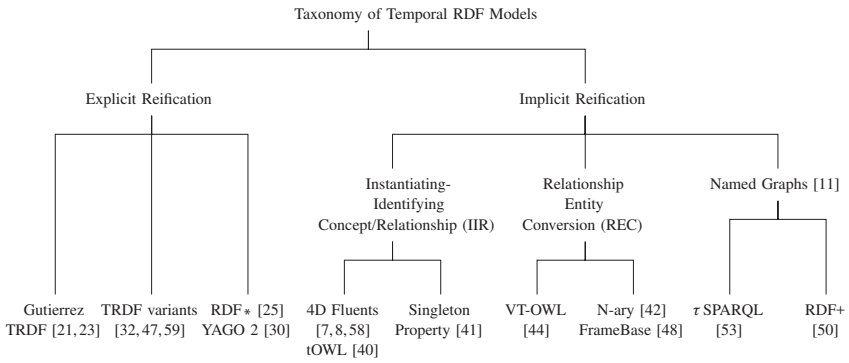

Figure 13 Taxonomy of temporal rdf models.

Table 3 Temporal model comparison

Approach	RDF/S Ext.	Add. Object	# of Triples	Semantics	Time Domain	Inst/ Interval	Query
Temporal RDF, Enh. YAGO2	N	Y	8 quadruple	RDF/S and Temporal graph closure	Model def.	Both	SWRL prototype
c-Temporal graph tRDF	N	Y	8	RDF/S, Temporal graph closure for indeterminate	Model def.	Both	Algorithm
RDF *	Y	Y	7	RDF/S	Model def.	Interval	SPARQL*
Annotated RDF	Y	Y	N/A[1]	Annotated logic and Inference rules	Model def.	Both	Algorithm
4D fluents and ext.	N	Y	7	OWL	OWL-time	Interval	SPARQL
tOWL	Y	Y	8	OWL-DL	Concrete domain	Both	SPARQL
Singleton property	Y	Y	5	RDF/S ext.	Instant	SPARQL	SPARQL
VT-OWL	N	Y	8	OWL	User	User	SPARQL
N-ary, framebase	N	Y	7	OWL	User	User	SPARQL
Named graphs	N	Y	3	RDF/S	User	User	SPARQL
τSPARQL	N	Y	3	RDF/S	OWL-time	Both	τSPARQL
RDF+	Y	Y	Quintuple	RDF+	Model def.	Both	Ext. SPARQL

[1]The annotated RDF framework does not discuss its model implementation.

5.1 Taxonomy

The temporal models for the Semantic Web surveyed in this paper use either explicit or implicit reification. In explicit reification, RDF reification vocabulary or its equivalent is used, whereas in implicit reification, some form of identification for a triple is introduced.

Temporal RDF [21, 23] and its variants [32, 47] are based on explicit RDF reification. RDF* introduces nested triples and needs an extension to RDF/S specifications. However, nested triples in a RDF* graph can be unnested to an explicitly reified RDF graph. Yago2 [30] reifies each fact and assigns an idenfier to it to form a quintuple. In contrast, all the other temporal models handle reification implicitly by using different forms of transformation on a triple, relationship or graph. Such a transformation does not rely on RDF/S reification vocabularies. The type of transformation differentiates these RDF models: (1) Instantiating-Identifying Concept/Relationship , (2) Relationship Entity Conversion and (3) Named Graphs.

There are two temporal models in Instantiating-Identifying Concept/Relationship models. 4D Fluents [7, 8, 58] introduces *temporal part* for an entity changing over time. Each temporal part corresponds to a distinguishable timestamp. A fluent property associates two temporal parts. On the other hand, Singleton Property [41] ensures every relationship to be universally unique. As a result, an ordinary relationship, such as *enrolled*, becomes a *relationship type*. Each of its instances, such as enrolled#1 in Figure 6, is used for an unique property assertion.

N-ary relations, FrameBase and OWL Temporal Model are examples of Relationship Entity Conversion models. N-ary relations [42] convert each relationship to an entity. FrameBase forms a semantic frame for each N-ary relation. Frame properties are asserted as semantic roles [18, 42]. OWL Temporal Model [44] is also based on N-ary relations. Lastly, Named Graphs are in RDF and RDFS specifications. A set of RDF/S triples can be identified with an IRI, that is, the graph name. Thus, graph level identification becomes available. As a result, additional properties for the graph can be associated through the graph's IRI.

5.2 RDF, RDFS, OWL Extension and Compliance

The majority of models extend RDF/S to gain temporal expressiveness. For instance, Temporal RDF [21, 23] and its variants [32, 47] introduce a set of temporal vocabularies to use explicit reification. The model provides a coverage of formal semantics of temporal graphs, query language prototype and complexity analysis. Similarly, 4D Fluents [7, 8, 58], N-ary relations [42] and Named Graphs [11] can all be implemented by RDF, RDFS and OWL vocabularies. These models benefit from available ontology tools, such as reasoners.

On the other hand, there are models that require formal extensions to RDF and RDFS specifications. For instance, Singleton Property [41] introduces *rdf:singletonPropertyOf* for instantiating a Singleton Property from its generic property type. Every Singleton Property is made universally unique. RDF* adopts *nested triples* to transform a triple to an entity. Nested triples informally allow *triple level identification* that is not available in RDF/S or OWL. As a result, RDF* requires syntactic and semantic extensions to RDF/S and OWL specifications. Similarly, Annotated RDFS [59] and RDF+ [50] require extensions to both RDF/S syntax and semantics. tOWL [40] extends OWL-DL to cover concrete domain for representing both time instants and intervals. Furthermore, it also incorporates Allen's temporal relations [5] to increase the model's temporal expressivity.

5.3 Additional Objects and Triples

When additional objects are required for a temporal model, they may cause quicker storage depletion or make writing standard queries more complex. The proliferation of objects is common in reification-based modeling approaches. In Temporal RDF model [21, 23], eight triples: four for reification and four for temporal assertions, are required to associate a time interval to an ordinary triple. 4D Fluents [7, 8, 58] use seven triples to cover temporal parts of an entity. Singleton Property [41] requires fewer triples (five triples). Named-Graphs [11] require only five temporal assertions, two triples for time instants, three for intervals. Since RDF+ has a form of quintuple [50] that can be mapped to standard RDF triples via explicit RDF reification, it requires at least eight triples.

5.4 Semantics

Typically, a temporal model extending RDF/S or OWL requires additional semantics, so temporal entailment can be defined. For instance, Temporal RDF [21, 23] specifies a temporal entailment semantics by using RDF and RDFS graphs. Additionally, this semantics extended to Enhanced Temporal RDF [32] and tRDF for Indeterminate Triples [47] with added temporal entailment scenarios, i.e., anonymous timestamp and indeterminate triples respectively.

In contrast, formal extensions to RDF/S or OWL semantics are introduced by extended vocabularies in other models. For instance, Annotated RDF [59] extends RDFS semantics by defining an algebraic structure for annotation domain, and also provides a deductive system. Singleton Property [41] requires RDF/S semantics extension to cover its *SingletonPropertyOf* interpretation. tOWL [40] requires a semantic extension for the translations between rational numbers, $\mathbb{Q}$, and XML datetime data types. τSPARQL [53] relies on Named Graphs where the identifier of a graph is a timestamp instead of an IRI. RDF+ [50] introduces additional semantics for its RDF+ literal and meta knowledge statement.

5.5 Time Domain

OWL-Time ontology includes class *TemporalEntity* [12] which is made-up of *Instants* and *Intervals*. Some of the temporal models use OWL-Time as their time domain. 4D fluents [58], extended 4D fluents [7, 8] and τSPARQL [53] all employ OWL-Time. The rest of the models do not use OWL-Time. They typically define a time domain or a time ontology of their own. For instance, the time domain in Temporal RDF [21, 23] is defined based on natural numbers. tOWL [40] uses the set of rational numbers, and provides a mapping to actual XML datetime type. Nevertheless, there are models that do not explicitly adopt a time domain specification. Instead, the time definition is left to applications.

5.6 Querying

SPARQL [24, 46] and SQWRL [43] or their extended forms are are used in querying temporal RDF data. SQWRL is for querying an OWL-based

ontology. Temporal RDF [21, 23] provide a query language sketch in rulelike form. An equivalent SPARQL query can be written directly, as it is based on RDF reification. Extensions to SPARQL syntax and semantics are needed for Singleton Property [41], RDF* [25], enhanced Temporal RDF [32, 47], RDF+ [50], τSPARQL [53], annotated RDF [59] and Extended 4D Fluents [7]. In general, such extension needs to accommodate additional patterns of syntax and query specifications. For instance, SPARQL* requires additional notation for nested triple while RDF+ adds *With Meta* and *From Named* constructs for querying its meta knowledge statements. A dedicated query language TOQL [6] is proposed for Extended 4D Fluents [7]. AnQL [36] is the query language tailored for Annotated RDFS [59]. SPARQL 1.1 directly supports Named Graphs, so there is no need for a new query language for Named Graphs [11].

6 Conclusion

Temporal models for the Semantic Web reported mainly extend RDF, RDFS or OWL to represent temporal data.These extensions use *Explicit Reification* and *Implicit Reification* which are the basis of a taxonomy we have developed to classify these models. While *Explicit Reification* is a method included in RDF standard, *Implicit Reification* aims at generating identity by which additional data can be specified in RDF. We expect that this taxonomy would be a base for better understanding of temporal RDF models.

Additionally, we have summarized various characteristics of these models and provided them in Table 3. Thus, Table 3 allows comparisons of the models and would be useful for the researchers and practitioners of the Semantic Web. A RDF, RDFS or OWL compliant temporal model can be implemented directly and also benefits from available tools, such as triple stores and reasoners. However, representing temporal data and knowledge definitely requires additional triples. The proliferation of triples causes performance, maintenance and storage issues. The ideal solution would be to reduce the number of additional triples to a minimum for representing temporal data. Also, this would make writing queries more intuitive and less complex.

There are several directions we plan to investigate. We will use the result of this survey to develop a new RDF model for representing temporal data that addresses the issues we have observed in the proposed models. We also plan to add other time dimensions to RDF. The majority of temporal models focuses on *valid-time* aspect of temporality. Nevertheless, being able to incorporate other temporal dimensions, such as transaction time, would allow richer implementation in temporal semantics in Web applications. A top-level time ontology that provides enough temporal expressivity, and facilitates more powerful temporal reasoning would be highly desirable. We also plan to investigate the possibility of how temporal support can be included in a top level ontology.

Acknowledgement

The authors would like thank anonymous reviewers for their valuable comments. Research of the second author is partially supported by the PSC-CUNY Award No. TRADA-48-475.

References

[1] IANA-Managed Reserved Domains. https://www.iana.org/domains/ reserved.

[2] Resource Description Framework (RDF). https://www.w3.org/RDF/.

[3] Web Ontology Language (OWL). https://www.w3.org/OWL/.

[4] Wikipedia: Manual of style/layout. https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Layout.

[5] James F. Allen. Maintaining Knowledge about Temporal Intervals. *Communication of the ACM*, 26:832–843, November 1983.

[6] Evdoxios Baratis, Euripides GM Petrakis, Sotiris Batsakis, Nikolaos Maris, and Nikolaos Papadakis. TOQL: Temporal Ontology Querying Language. In *International Symposium on Spatial and Temporal Databases*, pages 338–354. Springer, 2009.

[7] 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.

[8] Sotiris Batsakis, Euripides GM Petrakis, Ilias Tachmazidis, and Grigoris Antoniou. Temporal Representation and Reasoning in OWL 2. *Semantic Web*, 8(1):981–1000, 2017.

[9] Christian Bizer, Richard Cyganiak, Tobias Gauss, and Oliver Maresch. The TriQL.P Browser: Filtering Information using Context-, Content- and Rating-Based Trust Policies. In *Proceedings of the Semantic Web and Policy Workshop, held in conjunction with the 4th International Semantic Web Conference*, volume 7, pages 12–20, 2005.

[10] Gavin Carothers. RDF 1.1 N-Quads: A Line-Based Syntax for RDF Datasets. *W3C Recommendation*, 2014.

[11] Jeremy J Carroll, Christian Bizer, Pat Hayes, and Patrick Stickler. Named Graphs. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(4):247–267, 2005.

[12] Simon Cox, Chris Little, Jerry R Hobbs, and Feng Pan. *Time Ontology in OWL*. W3C recommendation, 2017.

[13] Richard Cyganiak, David Wood, Markus Lanthaler, Graham Klyne, Jeremy J Carroll, and Brian McBride. *RDF 1.1 Concepts and Abstract Syntax*. W3C recommendation, 2014.

[14] Vadim Ermolayev, Sotiris Batsakis, Natalya Keberle, Olga Tatarintseva, and Grigoris Antoniou. Ontologies of Time: Review and Trends. *International Journal of Computer Science & Applications*, 11(3), 2014.

[15] Charles J Fillmore, Christopher R Johnson, and Miriam RL Petruck. Background to Framenet. *International journal of lexicography*, 16(3):235–250, 2003.

[16] Flavius Frasincar, Viorel Milea, and Uzay Kaymak. tOWL: Integrating Time in OWL. In *Semantic Web Information Management*, pages 225–246. Springer Berlin Heidelberg, 2010.

[17] Antony Galton. Operators vs. Arguments: The Ins and Outs of Rreification. *Synthese*, 150(3):415–441, 2006.

[18] Aldo Gangemi and Valentina Presutti. A Multi-dimensional Comparison of Ontology Design Patterns for Representing n-ary Relations. In *SOFSEM*, volume 13, pages 86–105. Springer, 2013.

[19] Fabio Grandi. Multi-temporal RDF ontology versioning. In *Proceedings of the 3rd International Workshop on Ontology Dynamics (IWOD-09)*.

[20] TR Gruber. Toward Principles for the Design of Ontologies Used for Knowledge Sharing? *Pennsylvania: School of Information Sciences and Technology (IST). Pennsylvania State University. Acesso em*, 3, 1993.

[21] Claudio Gutierrez, Carlos Hurtado, and Alejandro Vaisman. Temporal RDF. In *European Semantic Web Conference*, pages 93–107. Springer, 2005.

[22] Claudio Gutierrez, Carlos A Hurtado, Alberto O Mendelzon, and Jorge Pérez. Foundations of Semantic Web Databases. *Journal of Computer and System Sciences*, 77(3):520–541, 2011.

[23] Claudio Gutierrez, Carlos A Hurtado, and Alejandro Vaisman. Introducing Time into RDF. *IEEE Trans. on Knowledge and Data Engineering*, 19:207–218, February 2007.

[24] Steve Harris, Andy Seaborne, and Eric Prudhommeaux. SPARQL 1.1 Query Language. *W3C Recommendation*, 21, 2013.

[25] Olaf Hartig and Bryan Thompson. Foundations of an Alternative Approach to Reification in RDF. In *Proceedings of the 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web*, 2017.

[26] Katherine Hawley. Temporal Parts. *Stanford Encyclopedia of Philosophy*, 2008.

[27] Patrick Hayes and Peter F. Patel-Schneider. *RDF 1.1 Semantics*. W3C recommendation, 2014.

[28] Patrick J Hayes. The Second Naive Physics Manifesto. 1985.

[29] Jerry R. Hobbs and Feng Pan. An Ontology of Time for the Semantic Web. *ACM Transactions on Asian Language Information Processing*, 3:66–85, 2004.

[30] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. *Artificial Intelligence*, 194:28–61, 2013.

[31] Ian Horrocks, Peter F Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean, et al. SWRL: A Semantic Web

Rule Language Combining OWL and RuleML. *W3C Member submission*, 21:79, 2004.

[32] Carlos A. Hurtado and Alejandro A. Vaisman. Reasoning with Temporal Constraints in RDF. In *PPSWR*, pages 164–178, 2006.

[33] Christian S Jensen, Curtis E Dyreson, Michael Böhlen, James Clifford, Ramez Elmasri, Shashi K Gadia, Fabio Grandi, Pat Hayes, Sushil Jajodia, Wolfgang Käfer, et al. The Consensus Glossary of Temporal Database ConceptsFebruary 1998 Version. In *Temporal Databases: Research and Practice*, pages 367–405. Springer, 1998.

[34] Michael Kifer and VS Subrahmanian. Theory of generalized annotated logic programming and its applications. *The Journal of Logic Programming*, 12(4):335–367, 1992.

[35] Adam Kilgarriff. Wordnet: An Electronic Lexical Database, 2000.

[36] Nuno Lopes, Axel Polleres, Umberto Straccia, and Antoine Zimmermann. AnQL: SPARQLing up Annotated RDFS. In *International Semantic Web Conference*, pages 518–533. Springer, 2010.

[37] Carsten Lutz. Adding Numbers to the SHIQ Description Logic–First Results. In *Proceedings of The 8th International Conference on Principles of Knowledge Representation and Reasoning (KR2002)*. Citeseer, 2001.

[38] John McCarthy. Situations, Actions, and Causal Laws. Technical report, STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1963.

[39] John McCarthy and Patrick J Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Readings in artificial intelligence*, pages 431–450, 1969.

[40] Viorel Milea, Flavius Frasincar, and Uzay Kaymak. tOWL: a Temporal Web Ontology Language. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(1):268–281, 2012.

[41] Vinh Nguyen, Olivier Bodenreider, and Amit Sheth. Don't like RDF Reification?: Making Statements about Statements using Singleton Property. In *Proceedings of the 23rd international conference on World wide web*, pages 759–770. ACM, 2014.

[42] Natasha Noy, Alan Rector, Pat Hayes, and Chris Welty. Defining n-ary Relations on the Semantic Web. *W3C Working Group Note*, 12(4), 2006.

[43] Martin O'Connor and Amar Das. SQWRL: A Query Language for OWL. In *Proceedings of the 6th International Conference on OWL: Experiences and Directions-Volume 529*, pages 208–215. CEUR-WS. org, 2009.

[44] Martin O'Connor and Amar Das. A Method for Representing and Querying Temporal Information in OWL. In *Biomedical Engineering Systems and Technologies, Communications in Computer and Information Science*. Springer, 2011.

[45] Bijan Parsia and Evren Sirin. Pellet: An OWL DL Reasoner. In *Third International Semantic Web Conference-Poster*, volume 18, 2004.

[46] Eric Prud, Andy Seaborne, et al. *SPARQL Query Language for RDF*. W3C Recommendation, 2006.

[47] Andrea Pugliese, Octavian Udrea, and VS Subrahmanian. Scaling RDF with Time. In *Proceedings of the 17th international conference on World Wide Web*, pages 605–614. ACM, 2008.

[48] Jacobo Rouces, Gerard de Melo, and Katja Hose. Framebase: Representing n-ary Relations Using Semantic Frames. In *European Semantic Web Conference*, pages 505–521. Springer, 2015.

[49] Anisa Rula, Matteo Palmonari, Andreas Harth, Steffen Stadtmüller, and Andrea Maurino. On the Diversity and Availability of Temporal Information in Linked Open Data. In *International Semantic Web Conference*, pages 492–507. Springer, 2012.

[50] Bernhard Schueler, Sergej Sizov, Steffen Staab, and Duc Thanh Tran. Querying for Meta Knowledge. In *Proceedings of the 17th international conference on World Wide Web*, pages 625–634. ACM, 2008.

[51] P Stickler. RDFQ, 2004.

[52] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a Core of Semantic Knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

[53] Jonas Tappolet and Abraham Bernstein. Applied Temporal RDF: Efficient Temporal Querying of RDF Data with SPARQL. In *European Semantic Web Conference*, pages 308–322. Springer, 2009.

[54] Dmitry Tsarkov and Ian Horrocks. FaCT++ Description Logic Reasoner: System description. In *International Joint Conference on Automated Reasoning*, pages 292–297. Springer, 2006.

[55] Octavian Udrea, Diego Reforgiato Recupero, and VS Subrahmanian. Annotated RDF. *ACM Transactions on Computational Logic (TOCL)*, 11(2):10, 2010.

[56] Frank van Harmelen and Deborah McGuinness. OWL Web Ontology Language Overview. W3C recommendation, W3C, February 2004. http://www.w3.org/TR/2004/REC-owl-features-20040210/.

[57] Max Volkel, Wolf Winkler, York Sure, S Ryszard Kruk, and Marcin Synak. Semversion: A versioning system for rdf and ontologies. In *Proc. of ESWC*, 2005.

[58] Chris Welty, Richard Fikes, and Selene Makarios. A Reusable Ontology for Fluents in OWL. In *Formal Ontology in Information Systems*, volume 150, pages 226–236, 2006.

[59] Antoine Zimmermann, Nuni Lopes, Axel Polleres, and Umberto Straccia. A General Framework for Representing, Reasoning and Querying with Annotated Semantic Web Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 11:72–95, 2012.

Biographies

Hsien-Tseng Wang received his BS and MS in Civil Engineering from the Central University in Taiwan. He also received an MBA from Baruch College, the City University of New York. He is currently a PhD candidate at the department of Computer Science at the Graduate Center, the City University of New York. Wang's research interests include machine learning, semantic analytics, temporal aspects of the Semantic Web and simulations.

Abdullah Uz Tansel received his BS in management, and his MS and PhD degrees in computer science from the Middle East Technical University, in Ankara Turkey. He has also received his MBA degree at the University of Southern California. After being a faculty member at the Middle East Technical University, Dr. Tansel joined Baruch College, the City University of New York (CUNY) where he is currently a professor of information systems and also a professor of computer science at The Graduate Center of CUNY. Professor Tansel's research focus is on temporal databases and he has made significant contributions in this field. He also headed the editorial board that published the first book on 'Temporal Databases: Theory, Design, and Implementation' (1993).

Dr. Tansel has a patent on adding temporality to RDF. His research interests are Database Management Systems, Temporal Databases, Semantic Web, and Blockchain Databases. Dr. Tansel has published many articles in the conferences and journals of the ACM, IEEE and other organizations. He is a frequent speaker on time in databases and blockchain as a database. He is also a member of the ACM and the IEEE Computer Society.