Temporally Enhanced Ontologies in OWL: A Shared Conceptual Model and Reference Implementation

Sven de Ridder and Flavius Frasincar

Erasmus University Rotterdam
PO Box 1738, NL-3000 DR
Rotterdam, the Netherlands

`svenderidder@gmail.com`, `frasincar@ese.eur.nl`

Abstract. The temporal dimension has been recognized as an integral feature of many Semantic Web applications, but there are significant differences in how ontology authors choose to represent changes in time. We present a temporal conceptual model for OWL DL ontologies that allows the expression of *fluent properties*, i.e., properties that change in time, that is both representation-agnostic and serializable for the various available representation schemes. We also provide *Kala*, a reference implementation developed in Java, that can be used to generate temporal ontologies, convert between temporal ontologies in different representation schemes, and develop new applications such as temporal querying or visualization tools.

Keywords: Semantic Web, OWL, time, fluents, Kala

1 Introduction

The ability to identify trends and make predictions is of critical importance for successful trading in financial markets. The increase in prominence of sophisticated, low-latency algorithmic trading systems has spurred development of technologies such as news analytics for the timely extraction of information that is relevant to the identification of market opportunities [24]. The Semantic Web, and OWL in particular, provide the technology to represent, manage, share, and reason over self-describing data, but these representations tend to be *synchronic*, i.e., they lack the crucial time dimension.

One reason for the lack of temporality in existing ontologies is that, while much effort has gone into providing support for temporal features at the representational level, there seems to be a lack of a shared, representation-agnostic model at the conceptual level, which is where user requirements commonly need to be met. Here, we consider the *conceptual level* to be the level at which humans may express and interpret information that closely relates to the perceived real world, the level that captures the essential semantics of temporal ontologies. In contrast, we consider the *representational level* to describe the organization of the information for representation and storage as computer data, typically as entities and relationships between entities, supported by a representation-specific vocabulary to express the individual data items. We shall refer to the conceptualization of temporal ontologies as the *temporal conceptual model*, and to the specification on the representational level as the *temporal representation scheme*.

Existing representation schemes are generally not directly compatible, and one result of the focus on representation schemes is that it greatly reduces the interoperability of various temporal implementations. Given the high conversion barriers, the ability to share and reuse data — a core objective of the Semantic Web — suffers. Secondly, the authors and users of temporal ontologies are directly exposed to the details of the particular representation scheme, which makes the development and use of temporal ontologies a cumbersome, complex, and error-prone process. Lastly, the focus on specific representation schemes results in applications that operate on temporal ontologies becoming tightly bound to a particular logical structure and implementation. This discourages the development of such applications, because their potential audience will be limited to the users of a particular representation scheme. Examples of these applications are temporally-enhanced reasoning, querying, and visualization.

Our focus will be on the introduction of concepts that form the building blocks of the semantics for temporally enhanced ontologies. The temporal conceptual model is designed to be mappable to selected representation schemes in OWL DL; that is, these representation schemes can be expressed in the $\mathcal{SHOIN}(\mathcal{D})$ description logic, and are fully compatible with the $\mathcal{SROIQ}(\mathcal{D})$ description logic employed by OWL 2. The model is, itself, composed of two orthogonal partitionings: one that describes the *time domain*, while the other describes *fluent properties*. Fluent properties, first described in the earliest literature on computer learning and artificial intelligence by McCarthy and Hayes [21], represent properties and relationships that may change with time.

Fluent properties form a suitable focal point for the exploration of a temporal conceptual model for a number of reasons. Firstly, the concept is immediately familiar: one does not have to stretch the imagination to think of examples of properties and relationships that change over time; a person's address, employer, and even favorite soccer team are all subject to such change — in fact, it may be more difficult to conceive examples of properties and relationships that categorically do *not* change over time! Secondly, fluent properties are conceptually simple: in effect, they represent ternary relations that simply extend the familiar binary relations with a fixed role for the third operand, the time interval. Thirdly, they are useful; as we have argued above, fluent properties allow for the evolution of an ontology to be expressed, examined, queried, and visualized. Lastly, fluent properties are already supported, in some form, through existing representation schemes.

This paper is structured as follows. Section 2 presents background on temporal models in general and the current state of temporal ontologies in particular. After this, in Sect. 3, we present the formal description of the proposed temporal conceptual model. An example implementation is provided in Sect. 4, followed by an evaluation of the implementation in Sect. 5. Lastly, we give our concluding remarks and identify possible future work in Sect. 6.

2 Representations of Temporality

The topic of data temporality has enjoyed great prolificacy: the scientific literature is rich in discussions of temporality, from philosophical treatises of time to discussions of temporal infrastructures and reasoning. Historically, this interest stems from the importance of time in many real-world applications, from logging and scheduling systems to biomedical databases and algorithmic trading.

Much of the early research on data temporality has focused on the field of temporal databases, a topic with similarities to temporal ontologies, and one that is considerably more mature. In fact, a striking resemblance to the current state of temporal ontologies may be gleaned from past reports on the field of temporal databases. Pissinou et al., in their report [27] on a 1992 ARPA/NSF workshop that was aimed specifically at identifying problems within the field of temporal database technology, conclude that the many different custom extensions to the relational model, each intended to serve very specific user needs regarding temporal support, and the resulting lack of a common terminology, infrastructure, and conceptual model for temporal databases, are primary reasons for reduced adoption of temporal database technology; similarly, we find that the field of temporal ontologies faces the same issues. The researchers and participants also identify the ad-hoc nature of many applications extended to include temporal information and the understandable resistance to replace existing applications with full-fledged temporal database technology as obstacles in the development and adoption of a standard for temporal databases, and conclude that there is a need for *open architectures* that allow for easy conversions between different representations.

In response to such findings, a consensus temporal query language specification, *TSQL2* [28], was developed, but the specification, despite strong initial ISO interest, failed to catch on: by the time that SQL:1999 was formally published, the specification had failed to meet the committee's requirements and could not be included in the language standard, and SQL vendor interest waned. Eventually, however, a number of key ideas from TSQL2 found their way into the SQL:2011 specifications [22]. Of these ideas, the concepts of *valid-time* ("application-time tables") and *transaction-time* ("transaction-time tables") have proved particularly useful: valid-time marking the time that a fact is held to be *true*, and transaction-time marking the time that a fact is *known* in the database. The approach to temporality in databases, then, typically resolved to marking tuples with valid-time and transaction-time timestamps, and this formed the basis for the general temporal database model (see, e.g., the conceptual model by Jensen et al. [16], the survey of temporal databases by Özsoyoğlu and Snodgrass [26], and the survey of temporal entity-relationship models by Gregersen and Jensen [11]).

With the development of the Semantic Web and its primary languages, the Resource Definition Language (RDF) and the Web Ontology Language (OWL), came efforts to represent temporal information in these languages. The Temporal RDF [12] language extension and its related query language T-SPARQL [10] form the main solution approach to introduce time to RDF. Unfortunately, Temporal RDF is not compatible with OWL DL because of its use of *RDF reification*. Compounding this problem is the snapshot-based entailment mechanism of Temporal RDF, which expands any temporal statement defined over a time interval into a series of temporal statements defined over each time instant contained by the interval, and the lack of available serializations, for example to RDF/XML.

Another approach is *ontology versioning* [17], in which "snapshots" of the ontology are created for each state of the ontology during its development. Unsurprisingly, this comes at the cost of significant data redundancy. Moreover, its support for particular classes of queries (e.g., "when is fact S true in the database?") is limited. However, the approach may also be used to model *transaction-time*, and may then be considered to be completely orthogonal to other (generally *valid-time*) approaches discussed here. The application of ontology versioning, therefore, may be appropriate in some cases where transaction-time needs to be modeled in addition to valid-time, but, perhaps, at low enough resolution so as to reduce the impact of data redundancy.

There have also been proposals to extend description logics with valid-time; see, e.g., the surveys by Artale and Franconi [2] and Lutz et al. [20]. Such temporal description logics are generally based on the $\mathcal{ALC}$ description logic [8]. These extensions are generally not compatible with OWL: the decidability of temporal description logics is compromised when the language is extended to the full description logics of $\mathcal{SHOIN}(\mathcal{D})$ for OWL DL or $\mathcal{SROIQ}(\mathcal{D})$ for OWL 2 [3]. Opting for temporal description logics would also mean giving up on the rich toolset developed for the OWL language, such as editors and reasoners.

Representation schemes for modeling temporality in OWL DL ontologies generally follow either the *reification*[1] approach or the *4D-fluents* approach. In the reification approach, a property instance is *reified*, that is, converted into a proper instance, and the original property's subject and object instances, or subject instance and datatype value, are then related to the newly reified relation through conventional property assertions to retain the information expressed by the original $\langle \text{subject}, \text{property}, \text{object} \rangle$ or $\langle \text{subject}, \text{property}, \text{value} \rangle$ triples. However, since we are now able to specify the reified property as the subject or object of additional triples, we effectively gain the ability to express properties that are ternary, quaternary, or generally *n-ary* in nature. The general approach of reification is, therefore, appropriately named *n-ary relations* [25].

At first glimpse, the reification approach seems appropriate for adding a ternary component, e.g., valid time, to any property assertion, and develop temporal ontologies based on temporally qualified properties. The reification approach is not without problems, however. One problem is the proliferation of

[1] Note that the reification representation scheme is not the same as the RDF reification, the latter being not available in OWL DL.

objects, namely one for each reified property assertion; related to this is the problem of providing meaningful names to the reified properties, or, alternatively, dealing with objects that may not have meaningful names. Another problem is the reduction of OWL reasoning capabilities over ontologies with reified properties; property semantics such as inverses or cardinalities are difficult or impossible to describe for reified properties in a general reasoning context.

In contrast to reification, the 4D-fluents approach [29] does not associate property assertions with valid-time intervals directly, but instead opts to have temporal properties hold between *timeslices* of entities, a timeslice being defined as the temporal facet of some entity as it "occupies" some interval in time. In order to be consistent, both subject and object timeslices must be *compatible*, that is, occupy the same interval in time. An important advantage of the 4D-fluents approach over the reification approach is that properties retain their semantics in reasoning contexts: for example, we may trivially define the inverse of a fluent property, as well as symmetry and transitivity; something that is not straightforward in the reification approach. The 4D-fluents approach, however, suffers from worse object proliferation than the reification approach in the general case.

The 4D-fluents approach has inspired several implementations. *tOWL* [23] employs the 4D-fluents approach and combines it with concrete domains and Allen's interval algebra [1] to allow the expression of temporal restrictions. *SOWL* [5], a spatio-temporal representation, uses the 4D-fluents approach as its temporal component. A reinterpretation of the 4D-fluents approach is implemented by the *MUSING* project [19], which focuses on adoption of the approach in the context of a reasoning architecture. Baratis et al. propose the 4D-fluents approach, combined with Allen's interval algebra, as the basis for *TOQL* [4].

Both approaches employ strategies that force conceptual concessions that conflict with intuitive understanding: the reification representation scheme models properties as classes, property assertions as instances, and prevents the user from specifying qualifiers for property semantics; the 4D-fluents representation scheme retains the property semantics, but requires the user to view instances as "spacetime-worms" and accept the conceptual implications that such a view necessitates. Converting a synchronic ontology with only static properties to a temporal ontology with dynamic properties is thus a cumbersome, error-prone process, as is the conversion between representation schemes. The lack of work on conceptual models for temporal ontologies in the literature indicates the need for improvements in this area.

3 The Temporal Conceptual Model

In this section we describe the proposed temporal conceptual model. Section 3.1 describes the time model. Section 3.2 builds on the time model to present the fluents model.

3.1 The Time Model

The time model extends the OWL model by introducing time instants and time intervals (the so-called *temporal primitives*), as well as assertions that relate these primitives to one another or assign to them discrete timestamp values. These temporal primitives and assertions form the building blocks for time models of varying expressive power and complexity.

The time model allows primitives to be declared explicitly through *primitive declarations*. Such declarations may or may not translate to OWL class membership declarations when serializing to a representation scheme R, depending on whether R represents time instants or time intervals as individuals or, instead, represents them directly as datatype values. The following axioms declare the anonymous individual `_:t1` to be a time instant, and the named individual `period2013Q1` to be an interval. We use a syntax that closely resembles the OWL *Abstract Syntax* in order to concisely express concepts in a familiar way.

```
TimeDeclaration(TimeInstant(_:t1))
TimeDeclaration(TimeInterval(period2013Q1))
```

The *"before"* relation between time instants t_1 and t_2 can be explicitly expressed in the temporal conceptual model, as shown below:

```
TimeInstantRelationAssertion(_:t1 _:t2 <)
```

Interval endpoint assertions relate time intervals to time instants: they specify that some time interval i_1 starts at a time instant t_s or ends at a time instant t_e. To preserve consistency, the assertion of interval endpoints t_s and t_e as, respectively, the start and the end time instants of time interval i implies that $t_s < t_e$ holds. The following axioms declare the time interval `period2013Q1` to start at time instants `_:t1` and `cal2013:jan1`, and end at time instant `_:t2`. Furthermore, the time interval `year2013` has the same start instant as `period2013Q1`. Note that `_:t1` and `cal2013:jan1` can be inferred to refer to the same time instant.

```
IntervalStartAssertion(period2013Q1 _:t1)
IntervalStartAssertion(period2013Q1 cal2013:jan1)
IntervalEndAssertion(period2013Q1 _:t2)
IntervalStartAssertion(year2013 _:t1)
```

Relations between time intervals are necessarily of a more complex nature than those between instants, because any two time intervals are not necessarily disjoint. Allen's work on interval relations [1] provides an algebra with useful qualities, and we have adopted the algebra to provide a mapping of relations between time intervals that is both jointly exhaustive and mutually exclusive. The time model allows interval relations to be expressed through the use of `TimeIntervalRelationAssertions`. As an example, consider the facts that `_:i1` *meets* (m) `_:i2`, and that `_:i1` *contains* (di) `_:i3`. The following statements assert these facts in the model:

```
TimeIntervalRelationAssertion(_:i1 _:i2 m)
TimeIntervalRelationAssertion(_:i1 _:i3 di)
```

In order to support timestamp values, we introduce *instant time assertions*. We have chosen to use the `xsd:dateTime` type to represent timestamp values, as it is already commonly used in OWL ontologies; however, the rather more complex time types introduced by the OWL-Time ontology [14] may also be considered in later versions. The `InstantTimeAssertion` represents the association of a time instant with a particular timestamp. For example, to express that `_:t1` is associated with June 5th, 2013, 6:42:23 PM in the Central European Summer Time (CEST: UTC+02:00) time zone, we declare the following assertion:

```
InstantTimeAssertion(_:t1
  "2013-06-05T18:42:23.000+02:00"^^xsd:dateTime)
```

3.2 The Fluents Model

The fluents model extends the OWL model by allowing the expression of *fluent properties*. These fluent properties resemble the standard OWL object and datatype properties, but assertions of these properties are additionally qualified with intervals from the time model previously introduced, indicating the (valid-time) interval that the fact is held to be true.

Like regular properties, fluent properties are *named resources*; that is, they can be referenced through identifiers that, in turn, may be extended to full URIs. They should not be considered OWL *entities*, however, for the simple reason that fluent properties are not part of the OWL specifications. It is important to note that the sets of regular properties and fluent properties are disjoint: no regular property may be used as a fluent property in the temporal conceptual model, and vice versa. One compelling reason for this separation is semantics: when a particular property is recognized to be a fluent property, its ability to change its value over time can be seen to be an intrinsic quality; declaring the property in a non-temporal context removes this ability and creates a contradiction. Another, more technical, reason is that neither the reification nor the 4D-fluents representation schemes allow for regular properties to be used as fluent properties, and vice versa: in the reification scheme, this would result in an OWL DL property and an OWL DL class sharing the same identifier, which is strictly prohibited by the standard; and in the 4D-fluents representation scheme, this would violate the domain and range restrictions on fluent properties: fluent datatype properties are restricted to domains of timeslice individuals, and fluent object properties are restricted to both domains and ranges of timeslice individuals.

The temporal conceptual model provides *fluent property declarations* to allow the ontology author to declare fluent properties explicitly. The manner in which these declarations are expressed in the eventual representation schemes is dependent on the details of the particular serialization. As an example, consider the following axioms, which declare a fluent object property `ceoOf` and a fluent datatype property `hasTitle`:

```
FluentsDeclaration(FluentObjectProperty(ceoOf))
FluentsDeclaration(FluentDataProperty(hasTitle))
```

The ontology author may wish to specify domain and range restrictions for fluent properties to restrict their use in ways that enforce correctness. These work similar to domain and range restrictions on regular properties: i.e., the domain of fluent object and datatype properties may be restricted to any *class expression*, as may the range of fluent object properties, and the range of fluent datatype properties may be restricted to any *data range*. The `hasTitle` fluent datatype property, for example, may be restricted as follows:

```
FluentDataPropertyDomain(hasTitle Person)
FluentDataPropertyRange(hasTitle DataOneOf(
  "Mr."^^xsd:string "Mrs."^^xsd:string "Ms."^^xsd:string))
```

As with regular properties, sequences of multiple domain or range restrictions on fluent properties are interpreted to represent the *intersection* of those domain or range restrictions.

A fluent object property assertion expresses the fact that a relation holds between two individuals during a particular time interval. Similarly, a fluent datatype property assertion expresses the fact that an attribute value holds for a particular individual during a particular time interval.

As fluent property assertions are at the heart of the temporal conceptual model, we shall provide a formal definition. Let us first define the concept of a *snapshot reduction* of a temporal ontology:

Definition 1. *Let $\mathcal{T} = \langle \mathcal{T}_p, < \rangle$, where $\mathcal{T}_p$ is a set of time instants and $<$ is a binary relation on the set $\mathcal{T}_p$ that is at least a strict partial order, be a linear, ordered time domain. Let $\mathcal{O}^{\mathcal{T}}$ be a temporally enhanced ontology over $\mathcal{T}$. Then, a* snapshot reduction *$\mathcal{O}^{\mathcal{T}}_t$ of $\mathcal{O}^{\mathcal{T}}$ at time $t \in \mathcal{T}_p$ is a non-temporal ontology that represents $\mathcal{O}^{\mathcal{T}}$ at time t.*

We can now formally define fluent object property assertions as follows:

Definition 2. *Let $\mathcal{C}$ be the set of all class expressions. Let $\mathcal{T} = \langle \mathcal{T}_p, < \rangle$, where $\mathcal{T}_p$ is a set of time instants and $<$ is a binary relation on the set $\mathcal{T}_p$ that is at least a strict partial order, be a linear, ordered time domain. Let $\mathcal{O}^{\mathcal{T}}$ be a temporally enhanced ontology over $\mathcal{T}$. Let $f^{OP} : D \longrightarrow R$, $D \subseteq \mathcal{C}$, $R \subseteq \mathcal{C}$ be a fluent object property, and let $OP : D \longrightarrow R$ be the non-temporal interpretation of f^{OP}. Then, a* fluent object property assertion*, $fa^{OP} = \langle s, f^{OP}, o, i \rangle$, $s \in D$, $o \in R$, $i = \langle t_s, t_e \rangle$, $t_s \in \mathcal{T}_p$, $t_e \in \mathcal{T}_p$, $t_s < t_e$ is said to* hold *between t_s and t_e if there exists an object property assertion $a^{OP} = \langle s, OP, o \rangle$ in every* snapshot reduction *$\mathcal{O}^{\mathcal{T}}_t$ of $\mathcal{O}^{\mathcal{T}}$, $t_s \leq t < t_e$.*

The formal definition of fluent datatype property assertions follows similarly, but is omitted for reasons of conciseness.

Using fluent property assertions, the ontology author may now express facts about individuals that only hold during a particular time interval. For example, in order to express that `sam` was the CEO of `ibm` during interval `_:i1`, and that `mary` was to be addressed as “Ms.” during interval `_:i2`, we add the following axioms:

```
FluentObjectPropertyAssertion(sam ceoOf ibm _:i1)
FluentDatatypePropertyAssertion(mary hasTitle
  "Ms."^^xsd:string _:i2)
```

4 Reference Implementation

To illustrate the use of our temporal conceptual model, we have developed *Kala* – Kālá being the Sanskrit word for time – as a proof-of-concept implementation. Kala is a Java Application Programming Interface (API) to aid the development of applications that need to create, manipulate, or query temporal ontologies at the conceptual level. It is based on the OWL API [15] and aims to have a design and interface that is familiar to users of that library. Like the OWL API, it provides an axiom-centric view of the (temporal) ontology, as opposed to the RDF triple-centric view of APIs such as Jena [6]. This axiom-centric view allows developers to utilize the library without concern for representation issues, in particular those related to parsing and serialization. Additionally, extending the OWL API will aid the later development of plug-ins for Protégé [18], a mature and widely-used ontology editor.

In order to guide the extension of the OWL API with temporal constructs, we pose two strong requirements:

1. The newly introduced temporal constructs must be completely separated from the constructs already supported by the OWL API in order to ensure the orthogonality of the non-temporal ontology and the temporal model; and
2. Kala must be compatible with custom implementations of the OWL API.

The orthogonality of the non-temporal components of the ontology and the temporal model is crucial for the correct functioning of the temporal model as an abstraction. We cannot, for example, decide that time instants and time intervals are subclasses of OWL individuals: time instants and time intervals are not necessarily represented as individuals in the representation scheme, and we need to restrict the operations that are permitted on these special entities in order to ensure that the temporal model will always have a correct mapping to the representation schemes. A statement such as, for example, “Bob is a friend of time instant t_1” does not make much sense in this context. Similar arguments hold for the fluent properties.

The compatibility with custom implementations of the OWL API is similarly crucial in the context of such developments as OWL database backends (for an examplar OWL database backend, see OWLDB [13]). In order to support custom implementations, we implement the extended functionality through the use of the *Decorator* design pattern [9]. We implement the additional Kala functionality by providing Decorators for the following three OWL API interfaces:

The ontology. The OWL API views an `OWLOntology`, essentially, as a collection of `OWLAxioms` and `OWLAnnotations`. It provides methods to query these axioms and annotations, directly or through convenience methods, and collaborates with other objects to change the contents of this internal collection. Introducing new categories of axioms, then, necessitates extending the `OWLOntology` in such a way that it can also store these new axioms, but without altering its existing behavior.

The data factory. The OWL API `OWLDataFactory` presents the interface for producing the entities, class expressions, and axioms that form the building blocks of the OWL ontology. The `OWLDataFactory` follows the *Factory* design pattern [9]. We extend the existing `OWLDataFactory` to support the construction of the entities and axioms of the temporal conceptual model.

The ontology manager. The `OWLOntologyManager`, lastly, is responsible for the creation, loading, saving, and manipulation of ontologies. Since we need to create a new type of ontology, the temporal ontology, we will need to extend the behavior of the `OWLOntologyManager` so that it can properly manage these temporal ontologies.

Parsing and serialization are performed through the `Parser` and `Serializer` interfaces, respectively. Each has a representation-scheme-specific specialization (e.g., `FluentsSerializer`), and all are initialized with data on the capabilities and vocabulary of the chosen representation through the `RepresentationScheme` object. The `Parser` follows the *Builder* design pattern [9]: it iterates through a provided `OWLOntology` object that represents the temporal ontology using a particular representation scheme, and builds and modifies structures that together represent the eventual `TemporalOntology` as it goes. The `Serializer`, on the other hand, is implemented as a *Visitor* that visits every `TemporalAxiom` in the ontology and serializes it as one or more OWL axioms. The insight here is that all `TemporalAxiom`s can be serialized independently of one another (something that is not true for *parsing*, where generally multiple OWL axioms must be parsed in combination in order to extract a single `TemporalAxiom`). The `Serializer` internally stores the set of generated axioms, and produces an `OWLOntology` from this set when the `createOntology()` method is called.

In order to raise awareness of the project and, hopefully, spur future developments, we have made the source code for Kala available on GitHub [7]. Due to space limitations, the details of the parsing and serialization algorithms can be found online at `http://tinyurl.com/qhw273a`.

5 Evaluation

We evaluate the temporal conceptual model — and Kala, our reference implementation — by representing a complex business process. The choice of a fitting scenario for our evaluation needs to satisfy three requirements. Firstly, the scenario must, naturally, convey temporal semantics that necessitate the utilization of a temporal model. Secondly, the complexity of the scenario must be sufficient to allow a range of expression types to be illustrated. Lastly, the scenario must present a relevance for the economic domain to prove its utility. One scenario fitting these requirements is found in the literature on tOWL. Illustrating the expressive power of tOWL, Milea et al. [23] present a historical account of the Leveraged Buy-Out (LBO) process for *Alliance Boots GmbH*, a multinational pharmaceuticals and retailing group that was formed through the 2006 merger of the *Boots group* and *Alliance UniChem*. In what was to be the largest LBO in European business history, two hedge funds, *Kohlberg Kravis Roberts & Co* (KKR) and *Terra Firma*, vied to acquire the company during the months of March and April of 2007, with the former emerging with the winning bid and, thus, the acquisition of Alliance Boots. Section 5.1 shows the modeling of this LBO process in the proposed conceptual model and Sect. 5.1 shows its serialization in two representation schemes.

5.1 Modeling the LBO Process

For reasons of conciseness, we refer the reader to the original paper [23] for the detailed representation of the LBO example, and limit ourselves to the definitions of fluent properties at the TBox level and the assertions of fluent properties and temporal entities at the ABox level. The authors identify four main stages of the LBO process: 1. Early Stage, 2. Due Diligence, 3. Bidding, and 4. Acquisition. These stages are pairwise disjoint in time and jointly exhaustive over the lifetime of the LBO process. Furthermore, the LBO process does not necessarily progress through the stages in a linear manner, and it may be aborted at any point in time.

TBox level. At the TBox level we find the conceptual information representing the `Company` type and its subtypes, `HedgeFund` and `Target`; the `Stage` type and its subtypes, `EarlyStage`, `DueDiligence`, and so on; and the various restrictions that enforce the validity of the representation. The TBox level is also where the fluent properties are defined. We present two such fluent properties: `earlyStage`, which temporally relates any `LBOProcess` to its `EarlyStage`; and `inStage`, which temporally relates any `Company` to any `Stage` of an `LBOProcess` in which it is involved. Finally, we present their respective restrictions. Using the syntax introduced in Sect. 3.2:

```
FluentsDeclaration(FluentObjectProperty(earlyStage))
FluentObjectPropertyDomain(earlyStage LBOProcess)
FluentObjectPropertyRange(earlyStage EarlyStage)

FluentsDeclaration(FluentObjectProperty(inStage))
FluentObjectPropertyDomain(inStage Company)
FluentObjectPropertyRange(inStage Stage)
```

ABox level. The ABox contains the assertional information related to any particular LBO process. Here, we find representations of participating companies, `alliance_boots`, `kkr`, and `terrafirma`; the instantiation of the LBO process we wish to represent, `lbo1`; and its various stages, `es1` and so on. Additionally, we represent the temporal relations between these entities, define the temporal primitives involved, and assign discrete time values. To represent, for example, the following news snippet of information:

Buyout firm Terra Firma mulls Boots bid
(Sun Mar 25, 2007 8:42 EDT)

we use the syntax introduced in Sects. 3.1 and 3.2:

```
TimeDeclaration(TimeInterval(i1))
TimeDeclaration(TimeInstant(t1))
IntervalStartAssertion(i1 t1)
InstantTimeAssertion(t1
  "2007-03-25T08:42:00-04:00"^^xsd:dateTime)

FluentObjectPropertyAssertion(lbo1 earlyStage es1 i1)
FluentObjectPropertyAssertion(alliance_boots inStage es1 i1)
FluentObjectPropertyAssertion(terrafirma inStage es1 i1)
```

5.2 Serialization to OWL DL representation schemes

We present the serializations of the temporal ontology discussed above to OWL DL representations of both the reification approach as well as the 4D-fluents approach below.

The reification representation. In the reification representation scheme, the declaration of a fluent property is serialized as a class declaration and property declarations for its subject and object relations. The serializer also specifies property restrictions for the subject and object relations and sets domain and range restrictions for the subject and object relations. An example for the `earlyStage` fluent object property is given below.

```
Class(earlyStage partial restriction(holds(someValuesFrom Interval)))
SubClassOf(earlyStage
  restriction(earlyStageToProcess(someValuesFrom LBOProcess)))
SubClassOf(earlyStage
  restriction(processToEarlyStage(someValuesFrom EarlyStage)))

ObjectProperty(earlyStageToProcess domain(earlyStage)
  range(LBOProcess))
Func(earlyStageToProcess)

ObjectProperty(processToEarlyStage domain(earlyStage)
  range(EarlyStage))
Func(processToEarlyStage)
```

Fluent property assertions are *reified* as anonymous individuals. Linking the subject, object, and interval to the fluent property is then done through a series of regular property assertions. The example below shows the serialization of linking the `lbo1` to its `es1` through a fluent object property assertion:

```
Individual(_:a1 type(earlyStage) value(earlyStageToProcess lbo1)
  value(processToEarlyStage es1) value(holds i1))
```

The 4D-fluents representation. The 4D-fluents representation differs from the original tOWL example [23] in a small number of ways. Firstly, the explicit class declarations for timeslice classes do not appear in the output. In the original serialization, for example, the class of all timeslices of the Early Stage is defined as in the following snippet (presented in OWL Abstract Syntax), which also shows the use of the explicit class description in the specification of an object property:

```
Class(EarlyStage_TS complete
  restriction(timeSliceOf(someValuesFrom EarlyStage)))

ObjectProperty(earlyStage
  domain(LBOProcess_TS)
  range(EarlyStage_TS))
```

This explicit class description is then used in axioms such as object property range restrictions. We do not generate such explicit class declarations for the timeslice classes and use the property restrictions directly in the expressions of such axioms, instead:

```
ObjectProperty(earlyStage
  domain(restriction(timeSliceOf(someValuesFrom LBOProcess))
  range(restriction(timeSliceOf(someValuesFrom EarlyStage)))
```

This alternate specification of the `earlyStage` property, however, conveys the same information as the original, but in a more compact manner.

At the ABox level, we introduce timeslices for every individual that participates in a fluent property. These timeslices are currently represented by *anonymous individuals* in the serialization, but otherwise follow the same serialization pattern as the original tOWL example. Timeslices are introduced for every *unique* combination of the individual participating in a fluent property and the time interval over which that fluent property holds. If an individual participates in multiple fluent properties over the same time interval, no additional timeslices are created, but instead, the same timeslice is reused for each fluent property. As an example, consider the timeslice relating to `lbo1`:

```
Individual(_:a1 type(TimeSlice) value(timeSliceOf lbo1)
  value(earlyStage _:a2) value(dueDiligence _:a3)
  value(bidding _:a4) value(abort _:a5))
```

6 Conclusion

In this paper, we have shown how the problem of defining the temporal conceptual model for OWL DL ontologies can be broken down into the definition of a *time model* that is *orthogonal* to the static ontology, and a *fluents model* that combines aspects of the static ontology and the time model to define the fluent properties. The concept of *orthogonality* was of key importance in allowing the concepts and expressivity of the time model and the fluents model to be established independently, which further allows the temporal conceptual model to be both simple and expressive. We have developed a reference implementation, Kala, with the goal of extending the OWL API with the additional constructs in a manner that closely resembles the design philosophy behind that library.

As future work, Kala could be extended in several ways. Adding the ability to produce a *snapshot reduction*, i.e., a regular OWL DL ontology that is representative of a temporal ontology at a particular time, would allow the temporal ontology to interact with existing tools such as OWL DL reasoners and query languages. Similarly, the access methods and internal data structures could be improved to provide greater convenience for users, for example by allowing the inspection of the evolution of a particular property's values over time. Such extensions could, finally, be used to develop applications on top of Kala. Tools such as temporal reasoners, temporal query languages, and temporal visualizations can benefit from Kala, as the functionality provided by these would represent a powerful argument to justify the use of the more complex temporal ontologies. For the ontology author, plugins for Integrated Development Environments such as Protégé, based on Kala, would allow the convenient development of temporal ontologies without considering the details of a particular representation scheme.

References

1. Allen, J.F.: Maintaining knowledge about temporal intervals. Communications of the ACM 26(11), 832–843 (1983)
2. Artale, A., Franconi, E.: A survey of temporal extensions of description logics. Annals of Mathematics and Artificial Intelligence 30(1-4), 171–210 (2000)
3. Artale, A., Lutz, C.: A correspondence between temporal description logics. Journal of Applied Non-Classical Logics 14(1-2), 209–233 (2004)
4. Baratis, E., Petrakis, E., Batsakis, S., Maris, N., Papadakis, N.: TOQL: temporal ontology querying language. In: 11th International Symposium on Advances in Spatial and Temporal Databases (SSTD 2009), pp. 338–354. Springer-Verlag, Berlin (2009)
5. Batsakis, S., Petrakis, E.G.M.: SOWL: spatio-temporal representation, reasoning and querying over the Semantic Web. In: 6th International Conference on Semantic Systems (I-Semantics 2010). p. 15. ACM Press, New York, NY (2010)
6. Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: implementing the Semantic Web recommendations. In: 13th International World Wide Web conference on Alternate track papers & posters. pp. 74–83. ACM Press, New York, NY (2004)
7. De Ridder, S.: owl-kala. `http://www.github.com/owl-kala/owl-kala`
8. Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A.: Reasoning in description logics. Principles of Knowledge Representation 1, 191–236 (1996)
9. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: abstraction and reuse of object-oriented design. Springer, Berlin (2001)
10. Grandi, F.: T-SPARQL: A TSQL2-like temporal query language for RDF. In: 1st International Workshop on Querying Graph Structured Data (GraphQ 2010) or-

ganized in conjunction with the 14th East-European Conference on Advances in Databases and Information Systems (ADBIS 2010). pp. 21–30 (2010)
11. Gregersen, H., Jensen, C.S.: Temporal entity-relationship models: a survey. IEEE Transactions on Knowledge and Data Engineering 11(3), 464–497 (1999)
12. Gutierrez, C., Hurtado, C.A., Vaisman, A.: Introducing time into RDF. IEEE Transactions on Knowledge and Data Engineering 19(2), 207–218 (2007)
13. Henß, J., Kleb, J., Grimm, S., Bock, J.: A database backend for OWL. In: 5th International Workshop on OWL: Experiences and Directions (OWLED 2009) (2009)
14. Hobbs, J.R., Pan, F.: An ontology of time for the Semantic Web. Transactions on Asian Language Information Processing 3(1), 66–85 (2004)
15. Horridge, M., Bechhofer, S.: The OWL API: A Java API for working with OWL 2 ontologies. In: 5th International Workshop on OWL: Experiences and Directions (OWLED 2009) (2009), `http://ceur-ws.org/Vol-529/owled2009_submission_29.pdf`
16. Jensen, C.S., Soo, M.D., Snodgrass, R.T.: Unifying temporal data models via a conceptual model. Information Systems 19(7), 513–547 (1993)
17. Klein, M., Fensel, D.: Ontology versioning on the Semantic Web. In: International Semantic Web Working Symposium (SWWS 2001). pp. 75–91. Springer-Verlag, Berlin (2001)
18. Knublauch, H., Fergerson, R.W., Noy, N.F., Musen, M.A.: The Protégé OWL plugin: An open development environment for Semantic Web applications. In: 3rd International Semantic Web Conference (ISWC 2004), pp. 229–243. Springer, Berlin (2004)
19. Krieger, H.U., Kiefer, B., Declerck, T.: A framework for temporal representation and reasoning in business intelligence applications. In: Hinkelmann, K. (ed.) AI Meets Business Rules and Process Management. Papers from AAAI 2008 Spring Symposium, pp. 59–70. AAAI Press, Stanford, CA (2008)
20. Lutz, C., Wolter, F., Zakharyashev, M.: Temporal description logics: A survey. In: 15th International Symposium on Temporal Representation and Reasoning (TIME 2008). pp. 3–14. IEEE Computer Society (2008)
21. McCarthy, J., Hayes, P.: Some Philosophical Problems from the Standpoint of Artificial Intelligence. Stanford University Press, Stanford, CA (1968)
22. Melton, J.: Information technology - database languages - SQL. part 2: Foundation (SQL/Foundation). Standard (ISO/IEC 9075-2:2011) (2011)
23. Milea, V., Frasincar, F., Kaymak, U.: tOWL: A temporal web ontology language. IEEE Transactions on Systems, Man and Cybernetics. Part B, Cybernetics (IEEE T-SMC-Part B) 42(1), 268–281 (2012)
24. Mitra, G., Mitra, L.: The handbook of news analytics in finance. John Wiley & Sons, Chichester, UK (2011)
25. Noy, N., Rector, A., Hayes, P., Welty, C.: Defining n-ary relations on the Semantic Web. W3C Working Group Note, April 2006, `http://www.w3.org/TR/swbp-n-aryRelations/` (2006)
26. Özsoyoğlu, G., Snodgrass, R.T.: Temporal and real-time databases: a survey. IEEE Transactions on Knowledge and Data Engineering 7(4), 513–532 (1995)
27. Pissinou, N., Snodgrass, R.T., Elmasri, R., Mumick, I.S., Özsu, T., et al.: Towards an infrastructure for temporal databases: report of an invitational ARPA/NSF workshop. ACM SIGMOD Record 23(1), 35–51 (1994)
28. Snodgrass, R.T.: The TSQL2 temporal query language. Kluwer, Boston, MA (1995)
29. Welty, C., Fikes, R., Makarios, S.: A reusable ontology for fluents in OWL. In: Bennett, B., Fellbaum, C. (eds.) 4th International Conference on Formal Ontology in Information Systems (FOIS 2006), pp. 226–236. IOS Press, Amsterdam (2006)