Modeling Concept Evolution: a Historical Perspective

Flavio Rizzolo, Yannis Velegrakis, John Mylopoulos, and Siarhei Bykau

University of Trento, Trento, 38100, Italy
{flavio, velgias, jm, bykau}@disi.unitn.eu

Abstract. The world is changing, and so must the data that describes its history. Not surprisingly, considerable research effort has been spent in Databases along this direction, covering topics such as temporal models and schema evolution. A topic that has not received much attention, however, is that of concept evolution. For example, Germany (instance-level concept) has evolved several times in the last century as it went through different governance structures, then split into two national entities that eventually joined again. Likewise, a caterpillar is transformed into a butterfly, while a mother becomes two (maternally-related) entities. As well, the concept of Whale (a class-level concept) changed over the past two centuries thanks to scientific discoveries that led to a better understanding of what the concept entails. In this work, we present a formal framework for modeling, querying and managing such evolution. In particular, we describe how to model the evolution of a concept, and how this modeling can be used to answer historical queries of the form "How has concept X evolved over period Y". Our proposal extends an RDF-like model with temporal features and evolution operators. Then we provide a query language that exploits these extensions and supports historical queries.

1 Introduction

Conceptual modeling languages – including the ER Model, UML class diagrams and Description Logics – are all founded on a notion of "entity" that represents a "thing" in the application domain. Although the state of an entity can change over its lifetime, entities themselves are atomic and immutable. Unfortunately, this feature prevents existing modeling languages from capturing phenomena that involve the evolution of an entity into something else, such as a caterpillar becoming a butterfly, or Germany splitting off into two Germanies right after WWII. In these cases, there is general agreement that an entity evolves into one or more different entities. Moreover, there is a strong relationship between the two, which is not captured by merely deleting one entity and then creating another. For example, the concept of Whale evolved over the past two centuries in the sense that whales were considered some sort of fish, but are now recognized as mammals. We are interested in modeling this kind of evolution relationship, not only at the instance and class levels but also across levels: instances may evolve into classes and viceversa. This notion of evolution is independent from the way the relationship between instances and classes is modeled [1].

In Databases, considerable amount of research effort has been spent on the development of models, techniques and tools for modeling and managing data changes. These range from data manipulation languages, and maintenance of views under changes [2],

to schema evolution [3] and mapping adaptation [4]. To cope with the history of data changes, temporal models have been proposed for the relational [5] and ER [6] models, for semi-structured data [7], XML [8] and for RDF [9]. Almost in its entirety, existing work on data changes is based on a data-oriented point of view. It aims at recording and managing changes that are taking place at the values of the data. What has been completely overlooked are other types of changes, such as an entity evolving/mutating into another, or an entity "splitting off" into several others.

In this work, we present a framework for modeling the evolution of concepts over time and the evolving relationships among them. The framework allows posing new kinds of queries that previously could not have been expressed. For instance, we aim at supporting queries of the form: How has a concept evolved over time? From what other concepts has it evolved and into what others has it resulted? What other concepts have affected its evolution over time? What concepts are indirectly related to it and how? These kinds of queries are of major importance for many interesting areas:

Historical Studies. Modern historians are interested in studying the history of human achievements, events and important persons. In addition, they want to understand how systems, tools, concepts, and techniques have evolved throughout the history. For them it is not enough to query a data source for a specific moment in history. They need to ask questions on how concepts and the relationships that exist between them have changed over time. Historians may be interested in the evolution of countries like Germany, with respect to territory, political division, etc. Alternatively, they may want to study the evolution of scientific topics, e.g., how the concept of biotechnology has evolved from its beginnings as an agricultural technology to the current notion that is coupled to genetics and molecular biology.

Entity Management. Web application and integration systems are progressively moving from tuple and value-based towards entity-based solutions, i.e., systems in which the basic data unit is an entity, independently of its logical modeling [10]. Furthermore, web integration systems, in order to achieve interoperability, may need to provide unique identification for the entities in the data they exchange [11]. Entities do not remain static over time: they evolve, merge, split, get created and disappear. Knowing the history of each entity, i.e., how it has been formed and from what, paves the ground for successful entity management solutions and affective information exchange.

Life Sciences One of the Biology fields is the study of the evolution of the species since life started on earth. To better understand the secrets of nature, it is important to model how the different species have evolved, from what, if and how they have disappeared, and when.

Our contributions are the following: (i) we consider a conceptual model enhanced with the notion of a lifetime of a class, individual and relationship, (ii) we further extend the temporal model [9] with consistency conditions and additional constructs to model merges, splits, and other forms of evolution among class-level and instance-level concepts; (iii) we introduce a query language that allows the answering of queries regarding the lifetime of concepts as well as the way they have evolved over time, along with other associated (via evolution) concepts; and finally, (iv) we present a case study in which we have applied our framework.

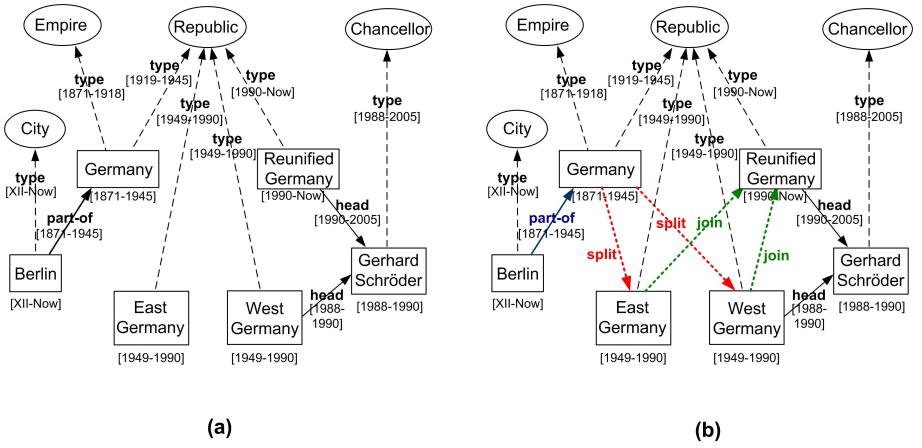

Fig. 1. The concepts of Germany, in a temporal model **(a)** and in our evolutionary model **(b)**

2 A Motivating Example

Consider the knowledge base of a historian that records information about countries and their political governance. A fraction of that information modeling a part of the history of Germany is illustrated in Figure 1 (a). In it, the status of Germany at different times in history has been been modeled through different individuals or through different instantiations. In particular, since 1871 and until 1945, Germany has been an empire and later a republic. This change is modeled by the multiple instantiation of Germany to Empire and Republic respectively. Shortly after the end of the World War II, Germany was split in four zones[1] that in 1949 formed the East and West Germany. These two parts lasted until 1990, when they were merged to form the republic of Germany as we know it today, which is modeled through the individual Reunified Germany.

To model the validity of each state of Germany at the different periods, a temporal model similar to temporal RDF [9] can be used. The model associates to each concept or individual a specific time frame. The time frames assigned to the individuals that model Germany are illustrated in Figure 1 through the intervals next to each individual. The same applies to every property, instance and subclass relationship. Note, for example, how the instantiation relationship of Germany to Empire has a temporal interval from 1871 to 1918, while the one to Republic has a temporal interval from 1918 to 1945. It is important for such a knowledge base to contain no inconsistencies, i.e., situations like having an instantiation relationship with a temporal interval bigger than the interval of the class it instantiates. Although temporal RDF lacks this kind of axiomatization, a mechanism for consistency checking needs to be in place for the accurate representation of concepts and individuals in history. Our solution provides an axiomatization of the temporal RDF that guarantees the consistency of the historical information recorded in a knowledge base.

[1] The information of the four zones is not illustrated in the Figure 1

Consider now the case of a historian that is interested in studying the history of Germany. Typical historian queries are those asking for all the leaders and constituents of a country from all its phases through time. Using traditional query mechanisms, the historian will only be able to retrieve the individual Germany. Using keyword based techniques, she may be able to also retrieve the remaining individuals modeling Germany at different times, but this under the assumption that each such individual contains the keyword "Germany". Applying terminology evolution [12], it is possible to infer that the four terms for Germany, i.e., Germany, East Germany, West Germany, and Reunified Germany refer to related concepts regardless of the keywords that appear in the terms. Yet, in neither case the historian will be able to reconstruct how the constituents of Germany have changed over time. She will not be able to find that the East and West Germany were made by splitting the pre-war Germany and its parts, neither that East and West Germany were the same two that merged to form the Reunified Germany. We propose the use of explicit constructs to allow the modeling of the sequential conceptual evolution in knowledge bases, as exemplified in Figure 1(b) by split, join and part-of.

3 Temporal Knowledge Bases

We consider an RDF-like data model. The model is expressive enough to represent ER models and the majority of ontologies and schemas that are met in practice [13]. It does not include certain OWL Lite features such as *sameAs* or *equivalentClass*, since these features have been considered to be out of the main scope of this work and their omission does not restrict the functionality of the model.

We assume the existence of an infinite set of *resources* $\mathcal{U}$, each with a *unique resource identifier* (URIs), and a set of *literals* $\mathcal{L}$. A *property* is a relationship between two resources. Properties are considered resources. We consider the existence of the special properties: rdfs:type, rdfs:domain, rdfs:range, rdfs:subClassOf and rdfs:subPropertyOf, which we denote for simplicity as type, dom, rng, subc, and subp, respectively. The set $\mathcal{U}$ contains three special resources: rdfs:Property, rdfs:Class and rdf:Thing, which we denote for simplicity as Prop, Class and Thing, respectively. The semantics of these resources as well as the semantics of the special properties are those defined in RDFS [14]. Resources are described by a set of triples that form a knowledge base.

Definition 1. *A* knowledge base Σ *is a tuple* $\langle U, L, T\rangle$*, where* $U \subseteq \mathcal{U}$*,* $L \subseteq \mathcal{L}$*,* $T \subseteq \mathcal{U} \times \mathcal{U} \times \{\mathcal{U} \cup \mathcal{L}\}$*, and* U *contains the resources* rdfs:Property*,* rdfs:Class*, and* rdf:Thing*. The set of* classes *of the knowledge base* Σ *is the set* $C=\{x \mid \exists\langle x, \mathsf{type}, \mathsf{rdfs{:}Class}\rangle \in T\}$*. Similarly, the set of* properties *is the set* $P=\{x \mid \exists\langle x, \mathsf{type}, \mathsf{rdfs{:}Property}\rangle \in T\}$*. The set* P *must contain the RDFS properties* type*,* dom*,* rng*,* subc*, and* subp*. A resource* i *is said to be an* instance *of a class* $c \in C$ *(or of type* c*) if* $\exists\langle i, \mathsf{type}, c\rangle \in T$*. The set of* instances *is the set* $I=\{i \mid \exists\langle i, \mathsf{type}, y\rangle \in T\}$.

A knowledge base can be represented as a hypergraph called an *RDF graph*. In the rest of the paper, we will use the terms knowledge base and RDF graph equivalently.

Definition 2. *An* RDF graph *of a knowledge base* Σ *is an hypergraph in which nodes represent resources and literals and the edges represent triples.*

Example 1. Figure 1(a) is an ilustration of an RDF Graph. The nodes Berlin and Germany represent resources. The edge labeled part-of between them represents the triple ⟨Berlin,part-of,Germany⟩. The label of the edge, i.e., part-of, represents a property.

To support the temporal dimension in our model, we adopt the approach of Temporal RDF [9] which extends RDF by associating to each triple a time frame. Unfortunately, this extension is not enough for our goals. We need to add time semantics not only to relationships between resources (what the triples represent), but also to resources themselves by providing temporal-varying classes and individuals. This addition and the consistency conditions we introduce below are our temporal extensions to the temporal RDF data model.

We consider time as a discrete, total order domain $\mathbb{T}$ in which we define different granularities. Following [15], a *granularity* is a mapping from integers to *granules* (i.e., subsets of the time domain $\mathbb{T}$) such that contiguous integers are mapped to non-empty granules and granules within one granularity are totally ordered and do not overlap. Days and months are examples of two different granularities, in which each granule is a specific day in the former and a month in the latter. Granularities define a lattice in which granules in some granularities can be aggregated in larger granules in *coarser* granularities. For instance, months are a coarser granularity than days because every granule in the former (a month) is composed of an integer number of granules in the latter (days). In contrast, months are not coarser (nor finer) than weeks.

Even though we model time as a point-based temporal domain, we use intervals as abbreviations of sets of instants whenever possible. An ordered pair $[a, b]$ of time points, with a, b granules in a granularity, and $a \leq b$, denotes the closed interval from a to b. As in most temporal models, the current time point will be represented with the distinguished word *Now*. We will use the symbol $\mathcal{T}$ to represent the infinite set of all the possible temporal intervals over the temporal domain $\mathbb{T}$, and the expressions $i.start$ and $i.end$ to refer to the starting and ending time points of an interval i. Given two intervals i_1 and i_2, we will denote by $i_1 \sqsubseteq i_2$ the containment relationship between the intervals in which $i_2.start \leq i_1.start$ and $i_1.end \leq i_2.end$.

Two types of temporal dimensions are normally considered: *valid* time and *transaction* time. Valid time is the time when data is valid in the modeled world whereas transaction time is the time when data is actually stored in the database. Concept evolution is based on valid time.

Definition 3. *A* temporal knowledge base *Σ_T is a tuple $\langle U, L, T, \tau \rangle$, where $\langle U, L, T \rangle$ is a knowledge base and τ is function that maps every resource $r \in U$ to a temporal interval in $\mathcal{T}$. The temporal interval is also referred to as the* lifespan *of the resource. The expressions $r.start$ and $r.end$ denote the start and end points of the interval of r, respectively. The* temporal graph *of Σ_T is the RDF graph of $\langle U, L, T \rangle$ enhanced with the temporal intervals on the edges and nodes.*

For a temporal knowledge base to be semantically meaningful, the lifespan of the resources need to satisfy certain conditions. For instance, it is not logical to have an individual with a lifespan that does not contain any common time points with the lifespan of the class it belongs to. Temporal RDF does not provide such a mechanism, thus, we are introducing the notion of a consistent temporal knowledge base.

Definition 4. *A* consistent temporal knowledge base *is a temporal knowledge base* $\Sigma_\tau = \langle U, L, T, \tau \rangle$ *that satisfies the following conditions:*

1. $\forall r \in L \cup \{\mathsf{Prop}, \mathsf{Class}, \mathsf{Thing}, \mathsf{type}, \mathsf{dom}, \mathsf{rng}, \mathsf{subc}, \mathsf{subp}\}$: $\tau(r) = [0, Now]$;
2. $\forall \langle d, p, r \rangle \in T : \tau(\langle d, p, r \rangle) \sqsubseteq \tau(d)$ *and* $\tau(\langle d, p, r \rangle) \sqsubseteq \tau(r)$;
3. $\forall \langle d, p, r \rangle \in T$ *with* $p \in \{\mathsf{type}, \mathsf{subc}, \mathsf{subp}\}$: $\tau(d) \sqsubseteq \tau(r)$.

Intuitively, literals and the special resources and properties defined in RDFS need to be valid during the entire lifespan of the temporal knowledge base, which is $[0, Now]$ (Condition 1). In addition, the lifespan of a triple needs to be within the lifespan of the resources that the triple associates (Condition 2). Finally, the lifespan of a resource has to be within the lifespan of the class the resource instantiates, and any class or property needs to be within the lifespan of its superclasses or superproperties (Condition 3).

4 Modeling Evolution

Apart from the temporal dimension that was previously described, two new dimensions need to be taken into consideration to successfully model evolution: the mereological and the causal.

Mereology [16] is a sub-discipline in philosophy that deals with the ontological investigation of the part-whole relationships. It is used in our model to capture the part-hood relationship between concepts in a way that is carried forward as concepts evolve. Such a relationship is modeled through the introduction of the special property part-of, which is reflexive, antisymmetric and transitive. A property part-of is defined from a resource x to a resource y if the concept modeled by resource x is part of the concept modeled by resource y. Note that the above definition implies that every concept is also a part of itself. When x is a part of y and $x \neq y$ we say that x is a *proper part* of y. Apart from this special semantics, part-of behaves as any other property in a temporal knowledge base. For readability and presentation reasons, we may use the notation $x \overset{\mathsf{part\text{-}of}}{\longrightarrow} y$ to represent the existence of a triple $<x,\mathsf{part\text{-}of},y>$ in the set T of a temporal knowledge base Σ_τ.

To capture the causal relationships, i.e., the interdependency between two resources, we additionally introduce the notion of becomes, which is an antisymmetric and transitive relation. For similar reasons as before, we may use the notation $x \overset{\mathsf{becomes}}{\longrightarrow} y$ to represent the fact that $(x, y) \in \mathsf{becomes}$. Intuitively, $x \overset{\mathsf{becomes}}{\longrightarrow} y$ means that the concept modeled by resource y originates from the concept modeled by resource x. We require that $\tau(x).end < \tau(y).start$.

To effectively model evolution, we introduce the notion of a *liaison*. A liaison between two concepts is another concept that keeps the former two linked together in time by means of part-of and becomes. In other words, a liaison is part of at least one of the concepts it relates and has some causal relationship to a part of the other.

Definition 5 (Liaison). *Let* A, B *be two concepts of a temporal knowledge base with* $\tau(A).start < \tau(B).start$, *and* x, y *concepts for which* $x \overset{\mathsf{part\text{-}of}}{\longrightarrow} A$ *and* $y \overset{\mathsf{part\text{-}of}}{\longrightarrow} B$. *A concept* x *(or* y*) is said to be a* liaison between A and B *if either* $x \overset{\mathsf{becomes}}{\longrightarrow} y$ *or* $x = y$.

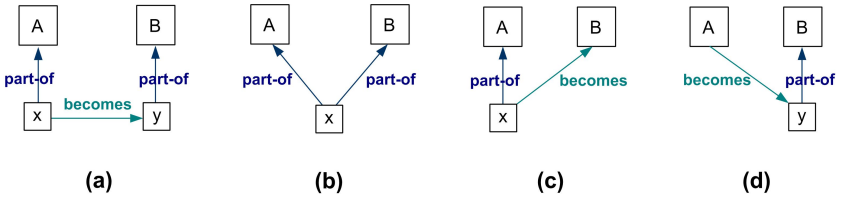

Fig. 2. Liaison examples

The most general case of a liaison is graphically depicted in Figure 2 (a). The boxes A and B represent the two main concepts whereas the x and y represent two of their respective parts. Figure 2 (b) illustrates the second case of the definition in which x and y are actually the same concept. Figure 2 (c) (respectively, (d)) shows the special case in which y (respectively, x) is exactly the whole of B (respectively, A) rather than a proper part of it.

To model the different kinds of evolution events that may exist, we introduce four evolution terms: join, split, merge, and detach.

[join] The join term, denoted as $\mathsf{join}(c_1 \ldots c_n, c, t)$, models the fact that every part of a concept c born at time t comes from a part of some concept in $\{c_1,\ldots,c_n\}$. In particular:

- $\tau(c).start{=}t$;
- $\forall x$ s.t. $x \xrightarrow{\mathsf{part\text{-}of}} c$: $\exists c_i$ s.t. x is a liaison between c_i and c, or $x = c_i$, with $1{\leq}i{\leq}n$.

[split] The split term, denoted as $\mathsf{split}(c, c_1 \ldots c_n, t)$, models the fact that every part of a concept c ending at time t becomes the part of some concept in $\{c_1,\ldots,c_n\}$. In particular:

- $\tau(c).end{=}t$;
- $\forall x$ s.t. $x \xrightarrow{\mathsf{part\text{-}of}} c$: $\exists c_i$ s.t. x is a liaison between c and c_i, or $x = c_i$, with $1{\leq}i{\leq}n$.

[merge] The merge term, denoted as $\mathsf{merge}(c, c', t)$, models the fact that at least a part of a concept c ending at a time t becomes part of an existing concept c'. In particular:

- $\tau(c).end{=}t$;
- $\exists x$ s.t. $x \xrightarrow{\mathsf{part\text{-}of}} c'$ and x is a liaison between c and c'.

[detach] The detach term, denoted as $\mathsf{detach}(c, c', t)$, models the fact that the new concept c' is formed at a time t with at least one part from c. In particular:

- $\tau(c').start{=}t$;
- $\exists x$ s.t. $x \xrightarrow{\mathsf{part\text{-}of}} c$ and x is a liaison between c and c'.

Note that in each evolution term there is only one concept whose lifespan has necessarily to start or end at the time of the event. For instance, we could use a join to represent the fact that different countries joined the European Union (EU) at different times. The information of the period in which each country participated in the EU is given by the interval of each respective part-of property.

We record the becomes relation and the evolution terms in the temporal knowledge base as *evolution triples* $\langle c, term, c' \rangle$, where $term$ is one of the special *evolution properties* becomes, join, split, merge, and detach. Evolution properties are *meta-temporal*, i.e., they describe how the temporal model changes, and thus their triples do not need to satisfy the consistency conditions in Definition 4. A temporal knowledge base with a set of evolution properties and triples defines an *evolution base*.

Definition 6. *An* evolution base Σ_T^E *is a tuple* $\langle U, L, T, E, \tau \rangle$*, where* $\langle U, L, T, \tau \rangle$ *is a temporal knowledge base,* U *contains a set of evolution properties, and* E *is a set of evolution triples. The* evolution graph *of* Σ_T^E *is the temporal graph of* $\langle U, L, T, \tau \rangle$ *enhanced with edges representing the evolutions triples.*

The time in which the evolution event took place does not need to be recorded explicitly in the triple since it can be retrieved from the lifespan of the involved concepts. For instance, detach(Kingdom of the Netherlands, Belgium, 1831) is modeled as the triple: $\langle$Kingdom of the Netherlands, detach, Belgium$\rangle$ with τ(Belgium)$.start = 1831$.

For recording evolution terms that involve more than two concepts, e.g. the join, multiple triples are needed. We assume that the terms are indexed by their time, thus, the set of (independent) triples that belong to the same terms can be easily detected since they will all share the same start or end time in the lifespan of the respective concept. For instance, split(Germany, East Germany, West Germany, 1949) is represented in our model through the triples $\langle$Germany, split, East Germany$\rangle$ and $\langle$Germany, split, West Germany$\rangle$ with τ(East Germany)$.start = \tau$(West Germany)$.start = 1949$.

Note that the evolution terms may entail facts that are not explicitly represented in the knowledge base. For instance, the split of Germany into West and East implies the fact that Berlin, which is explicitly defined as part of Germany, becomes part of either East or West. This kind of reasoning is beyond the scope of the current work.

5 Query Language

We define a navigational query language to traverse temporal and evolution edges in an evolution graph. This language is analogous to nSPARQL [17], a language that extends SPARQL with navigational capabilities based on nested regular expressions. nSPARQL uses four different axes, namely $\mathbf{self}$, $\mathbf{next}$, $\mathbf{edge}$, and $\mathbf{node}$, for navigation on an RDF graph and node label testing. We have extended the nested regular expressions constructs of nSPARQL with temporal semantics and a set of five *evolution axes*, namely $\mathbf{join}$, $\mathbf{split}$, $\mathbf{merge}$, $\mathbf{detach}$, and $\mathbf{becomes}$ that extend the traversing capabilities of nSPARQL to the evolution edges. The language is defined according to the following grammar:

$$exp := \mathbf{axis} \mid \mathbf{t\text{-}axis} :: a \mid \mathbf{t\text{-}axis} :: [exp] \mid exp[I] \mid exp/exp \mid exp|exp \mid exp^*$$

where a is a node in the graph, I is a time interval, and $\mathbf{axis}$ can be either $\mathbf{forward}$, $\mathbf{backward}$, $\mathbf{e\text{-}edge}$, $\mathbf{e\text{-}node}$, a $\mathbf{t\text{-}axis}$ or an $\mathbf{e\text{-}axis}$, with $\mathbf{t\text{-}axis} \in \{\mathbf{self}, \mathbf{next}, \mathbf{edge}, \mathbf{node}\}$ and $\mathbf{e\text{-}axis} \in \{\mathbf{join}, \mathbf{split}, \mathbf{merge}, \mathbf{detach}, \mathbf{becomes}\}$.

The evaluation of an evolution expression exp is given by the semantic function $\mathcal{E}$ defined in Figure 3. $\mathcal{E}[\![exp]\!]$ returns a set of tuples of the form $\langle x, y, I \rangle$ such that there

$$
\begin{aligned}
\mathcal{E}[\![\mathbf{self}]\!] &:= \{\langle x, x, \tau(x)\rangle \mid x \in U \cup L\} \\
\mathcal{E}[\![\mathbf{self}::r]\!] &:= \{\langle r, r, \tau(r)\rangle\} \\
\mathcal{E}[\![\mathbf{next}]\!] &:= \{\langle x, y, \tau(t)\rangle \mid t = \langle x, z, y\rangle \in T\} \\
\mathcal{E}[\![\mathbf{next}::r]\!] &:= \{\langle x, y, \tau(t)\rangle \mid t = \langle x, r, y\rangle \in T\} \\
\mathcal{E}[\![\mathbf{edge}]\!] &:= \{\langle x, y, \tau(t)\rangle \mid t = \langle x, y, z\rangle \in T\} \\
\mathcal{E}[\![\mathbf{edge}::r]\!] &:= \{\langle x, y, \tau(t)\rangle \mid t = \langle x, y, r\rangle \in T\} \\
\mathcal{E}[\![\mathbf{node}]\!] &:= \{\langle x, y, \tau(t)\rangle \mid t = \langle z, x, y\rangle \in T\} \\
\mathcal{E}[\![\mathbf{node}::r]\!] &:= \{\langle x, y, \tau(t)\rangle \mid t = \langle r, x, y\rangle \in T\} \\
\mathcal{E}[\![\mathbf{e\text{-}edge}]\!] &:= \{\langle x, \mathbf{e\text{-}axis}, [0, Now]\rangle \mid t = \langle x, \mathbf{e\text{-}axis}, z\rangle \in E\} \\
\mathcal{E}[\![\mathbf{e\text{-}node}]\!] &:= \{\langle \mathbf{e\text{-}axis}, y, \tau(t)\rangle \mid t = \langle z, \mathbf{e\text{-}axis}, y\rangle \in E\} \\
\mathcal{E}[\![\mathbf{e\text{-}axis}]\!] &:= \{\langle x, y, [0, Now]\rangle \mid \exists\, t = \langle x, \mathbf{e\text{-}axis}, y\rangle \in E\} \\
\mathcal{E}[\![\mathbf{forward}]\!] &:= \bigcup_{\mathbf{e\text{-}axis}} \mathcal{E}[\![\mathbf{e\text{-}axis}]\!] \\
\mathcal{E}[\![\mathbf{backward}]\!] &:= \bigcup_{\mathbf{e\text{-}axis}} \mathcal{E}[\![\mathbf{e\text{-}axis}^{-1}]\!] \\
\mathcal{E}[\![\mathbf{self}::[exp]]\!] &:= \{\langle x, x, \tau(x) \cap I\rangle \mid x \in U \cup L, \exists\langle x, z, I\rangle \in \mathcal{P}[\![exp]\!], \tau(x) \cap I \neq \emptyset\} \\
\mathcal{E}[\![\mathbf{next}::[exp]]\!] &:= \{\langle x, y, \tau(t) \cap I\rangle \mid t = \langle x, z, y\rangle \in T, \exists\langle z, w, I\rangle \in \mathcal{P}[\![exp]\!], \tau(t) \cap I \neq \emptyset\} \\
\mathcal{E}[\![\mathbf{edge}::[exp]]\!] &:= \{\langle x, y, \tau(t) \cap I\rangle \mid t = \langle x, y, z\rangle \in T, \exists\langle z, w, I\rangle \in \mathcal{P}[\![exp]\!], \tau(t) \cap I \neq \emptyset\} \\
\mathcal{E}[\![\mathbf{node}::[exp]]\!] &:= \{\langle x, y, \tau(t) \cap I\rangle \mid t = \langle z, x, y\rangle \in T, \exists\langle z, w, I\rangle \in \mathcal{P}[\![exp]\!], \tau(t) \cap I \neq \emptyset\} \\
\mathcal{E}[\![\mathbf{axis}^{-1}]\!] &:= \{\langle x, y, \tau(t)\rangle \mid \langle y, x, \tau(t)\rangle \in \mathcal{E}[\![\mathbf{axis}]\!]\} \\
\mathcal{E}[\![\mathbf{t\text{-}axis}^{-1}::r]\!] &:= \{\langle x, y, \tau(t)\rangle \mid \langle y, x, \tau(t)\rangle \in \mathcal{E}[\![\mathbf{t\text{-}axis}::r]\!]\} \\
\mathcal{E}[\![\mathbf{t\text{-}axis}^{-1}::[exp]]\!] &:= \{\langle x, y, \tau(t)\rangle \mid \langle y, x, \tau(t)\rangle \in \mathcal{E}[\![\mathbf{t\text{-}axis}::[exp]]\!]\} \\
\mathcal{E}[\![exp[I]]\!] &:= \{\langle x, y, I \cap I'\rangle \mid \langle x, y, I'\rangle \in \mathcal{E}[\![exp]\!] \text{ and } I \cap I' \neq \emptyset\} \\
\mathcal{E}[\![exp/e\text{-}exp]\!] &:= \{\langle x, y, I_2\rangle \mid \exists\langle x, z, I_1\rangle \in \mathcal{E}[\![exp]\!], \exists\langle z, y, I_2\rangle \in \mathcal{E}[\![e\text{-}exp]\!]\} \\
\mathcal{E}[\![exp/t\text{-}exp]\!] &:= \{\langle x, y, I_1 \cap I_2\rangle \mid \exists\langle x, z, I_1\rangle \in \mathcal{E}[\![exp]\!], \exists\langle z, y, I_2\rangle \in \mathcal{E}[\![t\text{-}exp]\!] \text{ and } \\
&\qquad I_1 \cap I_2 \neq \emptyset\} \\
\mathcal{E}[\![exp_1|exp_2]\!] &:= \mathcal{E}[\![exp_1]\!]\ cup \mathcal{E}[\![exp_2]\!] \\
\mathcal{E}[\![exp^*]\!] &:= \mathcal{E}[\![\mathbf{self}]\!] \cup \mathcal{E}[\![exp]\!] \cup \mathcal{E}[\![exp/exp]\!] \cup \mathcal{E}[\![exp/exp/exp]\!] \cup \ldots \\
\\
\mathcal{P}[\![e\text{-}exp]\!] &:= \mathcal{E}[\![e\text{-}exp]\!] \\
\mathcal{P}[\![t\text{-}exp]\!] &:= \mathcal{E}[\![t\text{-}exp]\!] \\
\mathcal{P}[\![t\text{-}exp/exp]\!] &:= \{\langle x, y, I_1 \cap I_2\rangle \mid \exists\langle x, z, I_1\rangle \in \mathcal{E}[\![t\text{-}exp]\!], \exists\langle z, y, I_2\rangle \in \mathcal{E}[\![exp]\!] \text{ and } \\
&\qquad I_1 \cap I_2 \neq \emptyset\} \\
\mathcal{P}[\![e\text{-}exp/exp]\!] &:= \{\langle x, y, I_1\rangle \mid \exists\langle x, z, I_1\rangle \in \mathcal{E}[\![e\text{-}exp]\!], \exists\langle z, y, I_2\rangle \in \mathcal{E}[\![exp]\!]\} \\
\mathcal{P}[\![exp_1|exp_2]\!] &:= \mathcal{E}[\![exp_1|exp_2]\!] \\
\mathcal{P}[\![exp^*]\!] &:= \mathcal{E}[\![exp^*]\!] \\
\\
t\text{-}exp &\in \{\mathbf{t\text{-}axis}, \mathbf{t\text{-}axis}::r, \mathbf{t\text{-}axis}::[exp], \mathbf{t\text{-}axis}[I]\} \\
e\text{-}exp &\in \{\mathbf{e\text{-}axis}, \mathbf{e\text{-}axis}::[exp], \mathbf{e\text{-}axis}[I], \mathbf{forward}, \mathbf{backward}\}
\end{aligned}
$$

Fig. 3. Formal semantics of nested evolution expressions

is a path from x to y satisfying exp during interval I. For instance, in the evolution base of Figure 1, $\mathcal{E}[\![\mathbf{self} :: \text{Germany}/\mathbf{next} :: \text{head}/\mathbf{next} :: \text{type}]\!]$ returns the tuple $\langle$Germany, Chancellor, $[1988, 2005]\rangle$. It is also possible to navigate an edge from a node using the **edge** axis and to have a nested expression $[exp]$ that functions as a predicate which the preceding expression must satisfy. For example, $\mathcal{E}[\![\mathbf{self}[\mathbf{next} :: \text{head}/\mathbf{self} :: \text{Gerhard Schröder}]]\!]$ returns $\langle$Reunified Germany, Reunified Germany, $[1990, 2005]\rangle$ and $\langle$West Germany, West Germany, $[1988, 1990]\rangle$.

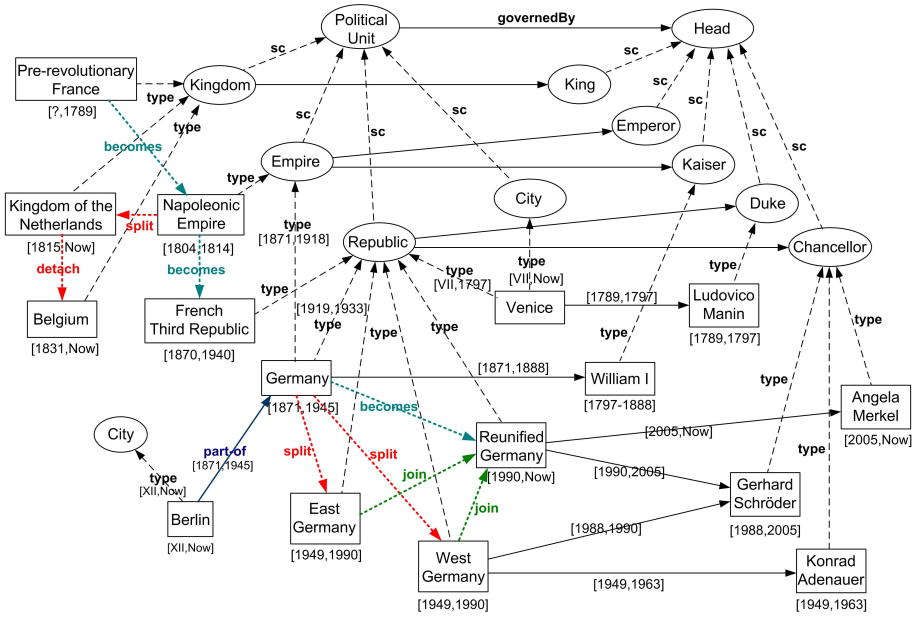

Fig. 4. The evolution of the concepts of Germany and France and their governments (full black lines represent governedBy properties)

In order to support evolution expressions, we need to extend nSPARQL triple patterns with temporal and evolution semantics. In particular, we redefine the evaluation of an nSPARQL triple pattern $(?X, exp, ?Y)$ to be the set of triples $\langle x, y, I\rangle$ that result from the evaluation of the evolution expression exp, with the variables X and Y bound to x and y, respectively. In particular:

$$\mathcal{E}[\![(?X, exp, ?Y)]\!] := \{(\theta(?X), \theta(?Y)) \,|\, \theta(?X) = x \text{ and } \theta(?Y) = y \text{ and } \langle x, y, I\rangle \in \mathcal{E}[\![exp]\!]\}$$

Our language includes all nSPARQL operators such as AND, OPT, UNION and FILTER with the same semantics as in nSPARQL. For instance:

$$\mathcal{E}[\![(P_1 \text{ AND } P_2)]\!] := \mathcal{E}[\![(P_1)]\!] \bowtie \mathcal{E}[\![(P_2)]\!]$$

where P_1 and P_2 are triple patterns and $\bowtie$ is the join on the variables P_1 and P_2 have in common. A complete list of all the nSPARQL operators and their semantics can be found in [17].

6 Application Scenarios

Consider an evolution base that models how countries have changed over time in terms of territory, political division, type of government, etc. Classes are represented by ovals and instances by boxes. A small fragment of that evolution base is illustrated as a graph in Figure 4.

Germany is a concept that has changed several times along history. The country was unified as a nation-state in 1871 and the concept of Germany first appears in our historical knowledge base as Germany at instant 1871. After WWII, Germany was divided into four military zones (not shown in the figure) that were merged into West and East Germany in 1949. This is represented with two split edges from the concept of Germany to the concepts of West Germany and East Germany. The country was finally reunified in 1990, which is represented by the coalescence of the West Germany and East Germany concepts into Unified Germany via two merge edges. These merge and split constructs are also defined in terms of the parts of the concepts they relate. For instance, a part-of property indicates that Berlin was part of Germany during $[1871, 1945]$. Since that concept of Germany existed until 1945 whereas Berlin exists until today, the part-of relation is carried forward by the semantics of split and merge into the concept of Reunified Germany. Consider now a historian who is interested in finding answers to a number of evolution-related queries.

[Example Query 1]: How has the notion of Germany changed over the last two centuries in terms of its constituents, government, etc.? The query can be expressed in our extended query language as follows:

$\mathbf{Select}\ ?Y, ?Z, ?W$
$(?X, \mathbf{self} ::\text{Reunified Germany}/\mathbf{backward}^*[1800, 2000]/, ?Y)$ AND
$(?Y, \mathbf{edge}, ?Z)$ AND $(?Z, \mathbf{edge}, ?W)$

The query first binds $?X$ to Reunified Germany and then follows all possible evolution axes backwards in the period $[1800, 2000]$. All concepts bound to $?Y$ are in an evolution path to Reunified Germany, namely Germany, West Germany, and East Germany. Note that, since the semantics of an * expression includes $\mathbf{self}$ (see Figure 3), then Reunified Germany will also bind $?Y$. The second triple returns in $?Z$ the name of the properties of which $?Y$ is the subject, and finally the last triple returns in $?W$ the objects of those properties. By selecting $?Y, ?Z, ?W$ in the head of the query, we get all evolutions of Germany together with their properties.

[Example Query 2]: Who was the head of the German government before and after the unification of 1990? The query can be expressed as follows:

$\mathbf{Select}\ ?Y$
$(?X, \mathbf{self} ::\text{Reunified Germany}/\mathbf{join}^{-1}[1990]/\mathbf{next} :: \text{head}[1990], ?Y)$ AND
$(?Z, \mathbf{self} ::\text{Reunified Germany}/\mathbf{next} :: \text{head}[1990], ?Y)$

The first triple finds all the heads of state of the Reunified Germany before the unification by following $\mathbf{join}^{-1}[1990]$ and then following $\mathbf{next}$ :: head$[1990]$. The second triple finds the heads of state of the Reunified Germany. Finally, the join on $?Y$ will bind the variable only to those heads of state that are the same in both triples, hence returning the one before and after the mentioned unification.

Consider now the evolution of the concept of biotechnology from a historical point of view. According to historians, biotechnology got its current meaning (related to molecular biology) only after the 70s. Before that, the term biotechnology was used in areas as diverse as agriculture, microbiology, and enzyme-based fermentation. Even

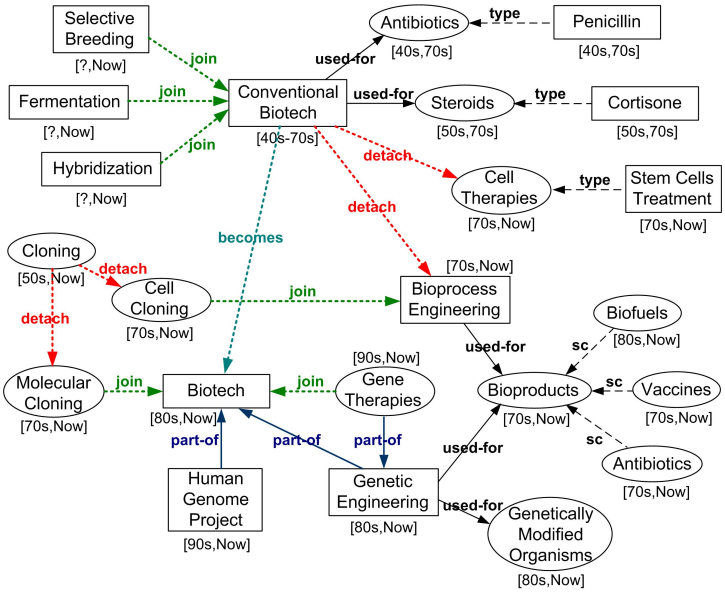

Fig. 5. The evolution of the concept of Biotechnology

though the term "biotechnology" was coined in 1919 by Karl Ereky, a Hungarian engineer, the earliest mentions of biotechnology in the news and specialized media refer to a set of ancient techniques like selective breeding, fermentation and hybridization. From the 70s the dominant meaning of biotechnology has been closely related to genetics. However, it is possible to find news and other media articles from the 60s to the 80s that use the term biotechnology to refer to an environmentally friendly technological orientation unrelated to genetics but closely related to bioprocess engineering. Not only the use of the term changed from the 60s to the 90s, but also the two different meanings coexisted in the media for almost two decades.

Figure 5 illustrates the evolution of the notion of biotechnology since the 40s. As in the previous example, classes in the evolution base are represented by ovals and instances by boxes. The used-for property is a normal property that simply links a technological concept to its products. The notions of Selective breeding, Fermentation and Hybridization existed from an indeterminate time until now and in the 40s joined the new topic of Conventional Biotech, which groups ancient techniques like the ones mentioned above. Over the next decades, Conventional Biotech started to include more modern therapies and products such as Cell Therapies, Penicillin and Cortisone. At some point in the 70s, the notions of Cell Therapies and Bioprocess Engineering matured and detached from Conventional Biotech becoming independent concepts. Note that Cell Therapies is a class-level concept that detached from the an instance-level concept. The three concepts coexisted in time during part of the 70s, the latter two coexist even now. During the 70s, the notion of Conventional Biotech stopped being used and all its related concepts became independent topics. In parallel to this, the new topic of Biotech

started to take shape. We could see Biotech as an evolution of the former Conventional Biotech but using Genetic Engineering instead of conventional techniques. Concepts and terms related to the Biotech and Genetic Engineering topics are modeled with a part-of property. In parallel to this, the concept of Cloning techniques started to appear in the 50s, from which the specialized notions of Cell Cloning and Molecular Cloning techniques detached in the 70s and joined the notions of Bioprocess Engineering and Biotech, respectively. This latter is an example of class-level concepts joining instance-level concepts.

[Example Query 3]: Is the academic discipline of biotechnology a wholly new technology branch or has it derived from the combination of other disciplines? Which ones and how? The query requires to follow evolution paths and return the traversed triples in addition to the nodes in order to answer the question of "how". The query is expressed in our language as follows:

$\mathbf{Select}\ ?Y, ?Z, ?W$
$(?X, \mathbf{self} :: \text{Biotechnology}/\mathbf{backward}^*, ?Y)$ AND
$(?Y, \mathbf{e\text{-}edge}/\mathbf{self}, ?Z)$ AND $(?Z, \mathbf{e\text{-}node}, ?W)$

The first triple binds $?Y$ to every node reachable from Biotechnology following evolution edges backwards. Then, for each of those nodes, including Biotechnology, the second triple gets all the evolution axes of which the bindings of $?Y$ are subjects whereas the third triple get the objects of the evolution axes. This query returns, (Biotech, $\mathbf{becomes}^{-1}$, Conventional Biotech), (Conventional Biotech, $\mathbf{join}^{-1}$, Hybridization), (Conventional Biotech, $\mathbf{join}^{-1}$, Fermentation), and (Conventional Biotech, $\mathbf{join}^{-1}$, Selective Breeding).

[Example Query 4]: Which scientific and engineering concepts and disciplines are related to the emergence of cell cloning? We interpret "related" in our model as being immediate predecessors/successors and "siblings" in the evolution process. That is, from a concept we first find its immediate predecessors by following all evolution edges backwards one step. We then follow from the result all evolution edges forward on step and we get the original concept and some of its "siblings". Finally, we repeat the same process in the opposite direction following evolution edges one step, first forward and then backwards. Based on this notion, we can express the query as follows:

$\mathbf{Select}\ ?Y, ?Z, ?W$
$(?X, \mathbf{self} :: \text{Cell Cloning}, ?Y)$ AND $?(Y, \mathbf{backward} \mid \mathbf{backward}/\mathbf{forward}, ?Z)$
AND $(?Y, \mathbf{forward} \mid \mathbf{forward}/\mathbf{backward}, ?W)$

The first triple will just bind Cell Cloning to $?Y$. The second triple follows the $\mathbf{detach}$ edge back to Cloning, and then the $\mathbf{detach}$ edge forward to Molecular Cloning. The third triple starts again from Cell Cloning and follows the $\mathbf{join}$ edge forward to Bioprocess Engineering and then the $detach$ edge backwards to Conventional Biotech. All these concepts will be returned by the query.

7 Related Work

Managing Time in Databases: Temporal data management has been extensively studied in the relational paradigm [5]. For semi-structured data, one of the first models for managing historical information as an extension of the Object Exchange Model (OEM) was introduced in [7]. A versioning scheme for XML was first proposed in [18]. Versioning approaches store the information of the entire document at some point in time and then use edit scripts and change logs to reconstruct versions of the entire document. In contrast, [19] and [8] maintain a single temporal document from which versions of any document fragment (even single elements) can be extracted directly when needed. A survey on temporal extensions to the Entity-Relationship (ER) model is presented in [6].

Change Management in Ontologies: There is a fundamental distinction between an actual update and a revision in knowledge bases [20]. An update brings the knowledge base up to date when the world it models has changed. Our evolution framework models updates since it describes how real-world concepts have changed over time. In contrast, a revision incorporates new knowledge from a world that has not changed. An approach to model revision in RDF ontologies has been presented in [21]. The survey in [22] provides a thorough classification of the types of changes that occur in ontologies. However, there is no entry in their taxonomy that corresponds to the kind of concept evolution we developed in this work; in fact, they view evolution as a special case of versioning. Similarly to versioning in databases, ontology versioning study the problem of maintaining changes in ontologies by creating and managing different variants of it [23]. Highly related, yet different, to concept evolution is the problem of terminology evolution that studies how terms describing the same notion in a domain of discourse are changing over time [12]. Closer to our work is the proposal in [24] for modeling changes in geographical information systems (GIS). They use the notion of a change bridge to model how the area of geographical entities (countries, provinces, etc.) evolve. A change bridge is associated with a change point and indicates what concepts become obsolete, what new concepts are created, and how the new concepts overlap with older ones. Since they focus on the GIS domain, they are not able to model causality and types of evolution involving abstract concepts beyond geographical entities.

8 Conclusion

In this work we studied the novel problem of concept evolution, i.e., how the semantics of an entity changes over time. In contrast to temporal models and schema evolution, concept evolution deals with mereological and causal relationships between concepts. Recording concept evolution also allows users to pose queries on the history of a concept. We presented a framework for modeling evolution as an extension of temporal RDF with mereology and causal properties expressed with a set of evolution terms. Furthermore, we presented an extension of nSPARQL that allows navigation over the history of the concepts. Finally, we applied our framework in two real world scenarios, the history of Germany and the evolution of biotechnology, and we showed how queries of interest can be answered using our proposed language.

Acknowledgments: The current work has been partially supported by the EU grants GA-215032 and ICT-215874.

References

1. Parsons, J., Wand, Y.: Emancipating instances from the tyranny of classes in information modeling. ACM Trans. Database Syst. **25**(2) (2000) 228–268
2. Blakeley, J., Larson, P.A., Tompa, F.W.: Efficiently Updating Materialized Views. In: SIGMOD. (1986) 61–71
3. Lerner, B.S.: A Model for Compound Type Changes Encountered in Schema Evolution. ACM Trans. Database Syst. **25**(1) (March 2000) 83–127
4. Velegrakis, Y., Miller, R.J., Popa, L.: Preserving mapping consistency under schema changes. VLDB J. **13**(3) (2004) 274–293
5. Soo, M.D.: Bibliography on Temporal Databases. SIGMOD Record **20**(1) (1991) 14–23
6. Gregersen, H., Jensen, C.S.: Temporal Entity-Relationship models - a survey. IEEE Trans. Knowl. Data Eng. **11**(3) (1999) 464–497
7. Chawathe, S., Abiteboul, S., Widom, J.: Managing historical semistructured data. In: Theory and Practice of Object Systems, Vol 5(3). (1999) 143–162
8. Rizzolo, F., Vaisman, A.A.: Temporal XML: modeling, indexing, and query processing. VLDB Journal **17**(5) (2008) 1179–1212
9. Gutiérrez, C., Hurtado, C.A., Vaisman, A.A.: Temporal RDF. In: ESWC. (2005) 93–107
10. Dong, X., Halevy, A.Y., Madhavan, J.: Reference Reconciliation in Complex Information Spaces. In: SIGMOD. (2005) 85–96
11. Palpanas, T., Chaudhry, J., Andritsos, P., Velegrakis, Y.: Entity Data Management in OKKAM. In: SWAP DEXA Workshop. (2008) 729–733
12. Tahmasebi, N., Iofciu, T., Risse, T., Niederee, C., Siberski, W.: Terminology evolution in web archiving: Open issues. In: International Web Archiving Workshop. (2008)
13. Lenzerini, M.: Data Integration: A Theoretical Perspective. In: PODS. (2002) 233–246
14. W3C: RDF vocabulary description language 1.0: RDF Schema. http://www.w3.org/TR/rdf-schema/ (2004)
15. Dyreson, C.E., Evans, W.S., Lin, H., Snodgrass, R.T.: Efficiently supported temporal granularities. IEEE Trans. Knowl. Data Eng. **12**(4) (2000) 568–587
16. Keet, C.M., Artale, A.: Representing and reasoning over a taxonomy of part-whole relations. Applied Ontology **3**(1-2) (2008) 91–110
17. Pérez, J., Arenas, M., Gutierrez, C.: nSPARQL: A navigational language for RDF. In: ISWC. (2008) 66–81
18. Chien, S., Tsotras, V., Zaniolo, C.: Efficient management of multiversion documents by object referencing. In: VLDB. (2001) 291–300
19. Buneman, P., Khanna, S., Tajima, K., Tan, W.: Archiving scientific data. In: SIGMOD. (2002) 1–12
20. Katsuno, H., Mendelzon, A.O.: On the difference between updating a knowledge base and revising it. In: KR. (1991) 387–394
21. Konstantinidis, G., Flouris, G., Antoniou, G., Christophides, V.: On RDF/S ontology evolution. In: SWDB-ODBIS. (2007) 21–42
22. Flouris, G., Manakanatas, D., Kondylakis, H., Plexousakis, D., Antoniou, G.: Ontology change: classification and survey. Knowledge Eng. Review **23**(2) (2008) 117–152
23. Klein, M.C.A., Fensel, D.: Ontology versioning on the semantic web. In: SWWS. (2001) 75–91
24. Kauppinen, T., Hyvönen, E.: Modeling and reasoning about changes in ontology time series. In: Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems. (2007) 319–338