Transforming XML to RDF(S) with Temporal Information

Dan Yang and Li Yan
College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China

The Resource Description Framework (RDF) is a model for representing resources on the Web. With the widespread acceptance of RDF in various applications (e.g., knowledge graph), a huge amount of RDF data is being proliferated. Therefore, transforming legacy data resources into RDF data is of increasing importance. In addition, time information widely exists in various real-world applications and temporal Web data has been represented and managed in the context of temporal XML. In this paper, we concentrate on transformation of temporal XML (eXtensible Markup Language) to temporal RDF data. We propose the mapping rules and mapping algorithms which can transform the temporal XML Schema and document into temporal RDF Schema and temporal RDF triples, respectively. We illustrate our mapping approach with an example and implement a prototype system. It is demonstrated that our mapping approach is valid.

Information systems → World Wide Web → Web data description languages → Markup languages → Extensible Markup Language (XML)
Information systems → Data management systems → Database design and models → Data model extensions → Temporal data

Keywords: RDF, temporal RDF, temporal XML, mapping

1. Introduction

TRDF (Resource Description Framework) [1] as well as RDF Schema (RDFS for short) recommended by the W3C (World Wide Web Consortium) can be applied as a metadata model for describing the semantics and reasoning about resources on the Web. RDF and/or RDFS are written as RDF(S). Nowadays RDF(S) has been widely accepted and used by many fields (e.g., governments, organizations and companies) as the data model and representation format, either for semantic data search and processing [28], or for representation of data from information extraction and integration. With the increasing amount of RDF data which is becoming available, transforming legacy data resources into RDF data is of increasing importance. Some efforts have been devoted to transformation of various data resources (e.g., relational databases [2] and UML class diagrams [3]) to RDF(S). Among various legacy data resources, XML (eXtensible Markup Language) [4] recommended by the W3C is the de-facto standard for data representation and exchange over the Web. XML has been extensively applied in many applications and a large volume of data is managed today directly in XML format. For this reason, some efforts have been made to transform XML into RDF(S) [5] – [11].

Time information widely exists in various real-world applications, including many web-based applications. Actually, many web data are typically time-related [12]. Some data on the Web, for example, are valid only at a certain time point or in a given time interval. In addition, some historical web data may need to be recorded and managed [13], [14]. In order to model and manage temporal information on the Web, temporal XML model has been proposed, because of the wide utilization of XML for data
representation and exchange on the Web (e.g., [15] – [20]). Note that XML cannot represent semantic information of data, although XML has been extensively used and a large volume of XML data is available. With the widespread acceptance and utilization of RDF as a meta- model for semantic data representation and processing on the Web, temporal RDF modeling has recently received more attention [22], [24], [25].

In this paper, we concentrate on modeling temporal information both in XML and RDF(S). In particular, we introduce the formal definitions of temporal XML model, including temporal XML Schema and temporal XML document, and temporal RDF(S) model. On this basis, we propose the formal approach to mapping the temporal XML model to the temporal RDF(S). We present the mapping rules and mapping algorithm. We apply an example and implement a prototype system to demonstrate that our mapping approach is valid. With the mapping approach proposed in the paper, temporal XML can be transformed into temporal RDF(S) and this serves as automation construction of temporal RDF(S) and further as semantic temporal information extraction and integration.

The rest of this paper is organized as follows. We present related work in Section 2. In Section 3, we introduce temporal data models we use in the paper, which are temporal XML model and temporal RDF model. We present our approach to mapping temporal XML model to temporal RDF model in Section 4, including the rules and algorithms of mapping temporal XML Schema to temporal RDF Schema and temporal XML document to temporal RDF triples, respectively. We design and implement a prototype in Section 5. Section 6 concludes this paper.

2. Related Work

The present work in this paper is closely related to two issues, which are classical XML transformation to RDF and temporal data modeling in XML and RDF.

2.1. Temporal Data Modeling

Temporal data modeling has been earlier investigated in the context of databases (e.g., [26]). With the wide utilization of XML for data representation and exchange on the Web, temporal XML modeling has received more attention. Time information is first considered in managing changes of semi-structured data [13], [14]. The time dimensions of temporal XML mainly include valid time and transaction time. In various temporal XML models proposed in literature, timestamps are explicitly added to nodes or/and edges of XML trees. In [15], two physical implementation models are presented for filling temporal data into XML documents. XML-based bitemporal data model (XBIT) proposed in [16] can manage valid time and transaction time histories. Without temporal information about XML Schema, the schemes for timestamps in XML documents are considered in [18]. A complex framework named eXSchema is presented in [17] to record the time-varying elements and physical implementation position. Temporal constraints are added to XML Schema in [19]. The issues of modeling, indexing, and querying temporal XML are investigated in [20].

Temporal RDF model is proposed to represent and manage temporal web data in a semantic way. In [21], [22], temporal reasoning is incorporated into RDF, yielding temporal RDF graphs. A temporal triple is defined as an RDF triple with a time interval. The syntax and deductive system for temporal RDF graphs are presented in [21], [22]. A very different temporal RDF model is proposed in [25], in which a temporal triple contains time information in its property component instead of a whole triple in [21], [22]. Several issues such as indexing [25], reasoning [23] and query [24] are investigated in the context of temporal RDF.

2.2. RDF Transformation from XML

Transformation of various data resources (e.g., relational databases [2] and UML class diagrams [3]) to classical RDF has been a crucial issue in RDF data management. In particular, some proposals establish the correspondences from XML to RDF(S). In [5], a procedure that transforms XML documents into RDF statements via an RDF-Schema specification is presented. A unified model for both XML and RDF by XML XQuery 1.0 and XPath 2.0 is provided in [6].

Transforming valid XML documents into RDF via RDF Schema is discussed by utilizing DTD (Document Type Definition) and XSD (XML Schema Definition) in [7] and [8], respectively. Considering the semantic similarity of duplicate elements in XML model (XSD or DTD), in [10] a set of rules is presented, which derives classes, properties, and data types from XML schema and interprets XML data as RDF statements by using RDF schema vocabularies. An approach for automatic transformation from XML to RDF via XML Schema is proposed in [9].

3. Temporal Data Models

We concentrate in this paper on the temporal XML and RDF data models containing two time-dimensions: valid time and transaction time. The valid time refers to the time when the data is true in the modeled reality and the transaction time denotes the time at which the information is edited. We apply two special time labels now and UC, to indicate the current time for valid and transaction, respectively. In addition, we apply the point-based time domain and encode some continuous time points with time intervals. We introduce two operations for time intervals, which are interval union and interval intersection (union and interval intersection, for short), respectively. Let \( t_1, t_2, \ldots \) be time intervals. Interval union, denoted \( \cup (t_1, t_2, \ldots) \), is applied to calculate the union of these time intervals and interval intersection, denoted \( \cap (t_1, t_2, \ldots) \), is applied to calculate the intersection of these time intervals.

In this section, we present the temporal models of 2 elements in an XML document, RDF, and RDF(S), respectively. We introduce the formal definitions of these temporal data models. Note that, for simplicity, only one time-dimension is presented at a formal definition and the other can be addressed in an analogous way.

3.1. Temporal XML Schema

XML Schema is used to define the structure of XML documents shared between applications. It always contains various elements and attributes with different structures and types. Following the step of [22], we extend the XML Schema model to support temporal information.

Definition 1. (temporal XML Schema model) A temporal XML Schema is a tuple \((\delta, T, M)\), in which

1. \( S = \{ S_1, \ldots, S_n \} \) is a set of XML Schemas and an XML Schema (say \( S \in S \)) is a tuple \((E_i, A_i, D, \rho, \kappa, \tau)\) [11], in which \( E_i \) is a set of elements, \( A_i \) is a set of attributes, and \( D \) is a set of simple datatypes which contains the built-in datatypes of XML Schema. The set of attributes of an element \( e \in E_i \) is defined by \( \rho: E_i \rightarrow D^* \) and \( \kappa: E_i \cup A_i \rightarrow D \) specifies the simple types of elements and attributes \( e \) (for the empty type).

2. \( T \) is a set of times.

3. \( M : S \rightarrow 2^T \) is a timestamp function that maps an XML Schema to a timestamp (a set of times).

Definition 2. (snapshot of temporal XML Schem- ma) Let \( TXS = (\delta, T, M) \) be a temporal XML Schema model. For \( t \in T \), snapshot \((t, TXS) = S_t \in S \), in which \( S_t \in S \) and \( t \in M(S) \).

Figure 1. An example of temporal XML Schema.
representation and exchange on the Web (e.g., [15] – [20]). Note that XML cannot represent semantic information of data, although XML has been extensively used and a large volume of XML data is available. With the widespread acceptance and utilization of RDF as a metadata model for semantic data representation and processing on the Web, temporal RDF modeling has received more attention [22], [24], [25].

In this paper, we concentrate on modeling temporal information both in XML and RDF(S). In particular, we introduce the formal definitions of temporal XML model, including temporal XML Schema and temporal XML document, and temporal RDF(S) model. On this basis, we propose the formal approach to mapping the temporal XML model to the temporal RDF(S). We present the mapping rules and mapping algorithm. We apply an example and implement a prototype system to demonstrate that our mapping approach is valid. With the mapping approach proposed in the paper, temporal XML can be transformed into temporal RDF(S), and this serves as automation construction of temporal RDF(S) and further as semantic temporal information extraction and integration.

The rest of this paper is organized as follows. We present related work in Section 2. In Section 3, we introduce temporal data models we use in the paper, which are temporal XML model and temporal RDF model. We propose our approach to mapping temporal XML model to temporal RDF model in Section 4, including the rules and algorithms of mapping temporal XML Schema to temporal RDF Schema and temporal XML document to temporal RDF triples, respectively. We design and implement a prototype in Section 5. Section 6 concludes this paper.

2. Related Work

The present work in this paper is closely related to two issues, which are classical XML transformation to RDF and temporal data modeling in XML and RDF.

2.1. Temporal Data Modeling

Temporal data modeling has been earlier investigated in the context of databases (e.g., [26]).

With the wide utilization of XML for data representation and exchange on the Web, temporal XML modeling has received more attention. Time information is considered in managing changes of semi-structured data [13], [14]. The time dimensions of temporal XML mainly include valid time and transaction time. In various temporal XML models proposed in literature, timestamps are explicitly added to nodes or/and edges of XML trees. In [15], two physical implementation models are presented for filling temporal data into XML documents. XML-based bitemporal data model (XBIT) proposed in [16] can manage valid time and transaction time histories. Without temporal information about XML Schema, the schemes for timestamps in XML documents are considered in [18]. A complex framework named rXSchema is presented in [17] to record the time-varying elements and physical implementation position. Temporal constraints are added to XML Schema in [19]. The issues of modeling, indexing, and querying temporal XML are investigated in [20].

Temporal RDF model is proposed to represent and manage temporal web data in a semantic way. In [21], [22], temporal reasoning is incorporated into RDF, yielding temporal RDF graphs. A temporal triple is defined as an RDF triple with a time interval. The syntax and deductive system for temporal RDF graphs are presented in [21], [22]. A very different temporal RDF model is proposed in [25], in which a temporal triple contains time information in its property component instead of a whole triple in [21], [22]. Several issues such as indexing [25], reasoning [23] and query [24] are investigated in the context of temporal RDF.

2.2. RDF Transformation from XML

Transformation of various data resources (e.g., relational databases [2] and UML class diagrams [3]) to classical RDF has been a crucial issue in RDF data management. In particular, some proposals establish the correspondences from XML to RDF(S). In [5], a procedure that transforms XML documents into RDF statements via an RDF-Schema specification is presented. A unified model for both XML and RDF by XML XQuery 1.0 and XPath 2.0 is provided in [6].

Transforming valid XML documents into RDF via RDF Schema is discussed by utilizing DTD (Document Type Definition) and XSD (XML Schema Definition) in [7] and [8], respectively. Considering the semantic similarity of duplicate XML data, using XML Schema and temporal information is presented in [9].

3. Temporal Data Models

We concentrate in this paper on the temporal XML and RDF data models containing two time-dimensions: valid time and transaction time. The valid time refers to the time when the data is true in the modeled reality and the transaction time denotes the time at which the information is edited. We apply two special time labels: now and UTC, to indicate the current time for valid and transaction, respectively. In addition, we apply the point-based time domain and encode some continuous time points with time intervals. We introduce two operations for time intervals, which are interval union and interval intersection (union and interval intersection, for short), respectively. Let \( t_1, t_2, \ldots \) be time intervals. Interval union, denoted \( \cup(t_1, t_2, \ldots) \), is applied to calculate the union of these time intervals and interval intersection, denoted \( \cap(t_1, t_2, \ldots) \), is applied to calculate the intersection of these time intervals.

In this section, we present the temporal models of 2 elements in XML document, RDF, and RDF(S), respectively. We introduce the formal definitions of these temporal data models. Note that, for simplicity, only one time-dimension is presented at a formal definition and the other can be addressed in an analogous way.

3.1. Temporal XML Schema

XML Schema is used to define the structure of XML documents shared between applications. It always contains various elements and attributes with different structures and types. Following the step of [22], we extend the XML Schema model to support temporal information.

Definition 1. (temporal XML Schema model) A temporal XML Schema model is a tuple \((\mathcal{S}, T, M)\), in which

\[ \mathcal{S} = \{S_1, \ldots, S_n\} \]

is a set of XML Schemas and an XML Schema (say \( S \in \mathcal{S} \)) is a tuple \((E, A, E_i, \rho, \kappa, \tau)\) [11], in which \( E \) is a set of elements, \( A \) is a set of attributes, and \( D \) is a set of simple datatypes which contains the built-in datatypes of XML Schema. The set of attributes of an element \( \epsilon \in E \) is defined by \( \rho: E \to 2^A \) and \( \kappa: E \cup A \to D \) specifies the simple types of elements and attributes \( \epsilon \) is for the empty type). Specifically, an element type may be more complex and can be presented by \( \tau = \varepsilon \) Order \( \{e_1, e_2, \ldots, e_n\} \in E \) where Order is an order indicator for (Sequence, Choice, All) of XML Schema.

\[ T \]

is a set of times.

\[ M: \mathcal{S} \to 2^T \]

is a timestamp function that maps an XML Schema to a timestamp (a set of times).

Definition 2. (snapshot of temporal XML Schema model) Let \( TXS = (\mathcal{S}, T, M) \) be a temporal XML Schema model. For \( t \in T \), snapshot \( (t, TXS) = S_t \in S \) in which \( S_t \in S \) and \( t \in M(S_t) \).

\[ \text{Figure 1. An example of temporal XML Schema.} \]
The notion of temporal XML Schema is applied to record different versions of XML schemas. Figure 1 depicts an example of temporal XML Schema. Here element SchemaVersion is used to record the version information of XML Schema and its children can be elements or other standard notations of XML Schema. Each child contains an attribute target which is used to record the location of the corresponding XML Schema via XPath. Note that, in the temporal Schema, the specified paths are the time intervals.

In order to accommodate temporal information in XML Schema, it is necessary to modify the corresponding XML Schema so that it can support temporal XML documents. For this purpose, we introduce several predefined structures which can be quoted to mark time varying contents. Note that these predefined structures can be contained in other schemas.

The first structure is an attributeGroup named temporalAttr, which contains four temporal attributes vStart, vEnd, tStart, and tEnd, representing the starting times and ending times of a valid time and a transaction time, respectively (Figure 2).

The second structure is a complexType named temporalAttrType, which is used by time varying attribute element (Figure 3).

The third structure is an element named temporalText, which can be contained by such elements that their texts are changing with time.

3.2. Temporal XML Documents

An XML document tree contains four kinds of nodes, which are root, element, attribute and text nodes, respectively. Actually, a root node is a special element node, which represents the beginning of an XML document uniquely. Connecting nodes forms the edges of other trees.

A temporal XML document is also represented as an XML tree. As shown in Figure 5, we attach bi-temporal annotations to tree nodes rather than to tree edges. Note that the valid or transaction time of a child node is contained by its parent node.

**Definition 3.** (Temporal XML document model). A temporal XML document model is a tuple $(X, T, M)$ [19], where

1. $X = \{X_1, \ldots, X_n\}$ is a set of XML documents and an XML document (say $X_i \in X$) is tuple $(N_i, ED_i, A)$. Here $N_i$ is a set of nodes (including all types of nodes in XML tree); $ED_i = \{(n_1, n_2)\} (n_1, n_2 \in N_i)$ is a set of edges; $A$ maps the nodes of the XML document model to an element set $E$ and an attribute set $A$ of an XML Schema.
2. $T$ is a set of times.
3. $M: X \rightarrow 2^T$ is a timestamp function that maps an XML document to a timestamp (a set of times).

**Definition.** (snapshot of temporal XML document). Let $TX = (X, T, M)$ be a temporal XML document model. For $t \in T$, snapshot $(t, TX) = X_t$, in which $X_t \in X$ and $t \in M(X_t)$.

An element of temporal XML documents is assigned with four special attributes vStart, vEnd, tStart and tEnd, which represent the valid and transaction interval of this element, respectively. When the attribute or text of an element is varying with time, we denote the element using a special attribute isAttr and represent the attribute or text with a subelement, in which the text node is denoted by an element named TextNode.

Let us look at an example. Suppose that we have a temporal element named phone, which has a text node which changes with time. Then we have a fragment of temporal XML document shown in Figure 6.

### 3.3. Temporal RDF(S)

RDF can be used to express propositions using precise formal vocabularies. W3C has given the semantics of RDF with model theory, which is regarded as “interpretation theory” in [27]. To represent time information in RDF, we need to add temporal information to this theory. In this paper, we adopt the temporal RDF model proposed in [22] with both valid and transaction dimensions by labeling.

**Definition 4.** (Temporal RDF model). A simple temporal interpretation of RDF is a tuple $(I, T, M)$, in which

<phone isAttr="True" vstart="2008/03/23" vend="now" tstart="2008/07/12" tend="UC">023-546887</phone>

This is a node text changing with time.

<TextNode>
</TextNode>
The notion of temporal XML Schema is applied to record different versions of XML schemas. Figure 1 depicts an example of temporal XML Schema. Here element SchemaVersion is used to record the version information of XML Schema and its children can be elements or other standard notations of XML Schema. Each child contains an attribute target which is used to record the location of the corresponding XML Schema via XPath. Note that, in the temporal Schema, the specified paths are the time intervals.

In order to accommodate temporal information in XML Schema, it is necessary to modify the corresponding XML Schema so that it can support temporal XML documents. For this purpose, we introduce several predefined structures which can be quoted to mark time varying elements. Note that these predefined structures can be contained in other schemas.

The first structure is an attributeGroup named temporalAttr, which contains four temporal attributes vStart, vEnd, tStart, and tEnd, representing the starting times and ending times of a valid time and a transaction time, respectively (Figure 2).

The second structure is a complexType named temporalAttrType, which is used by time varying attribute element (Figure 3).

The third structure is an element named temporalText, which can be contained by such elements that their texts are changing with time.

3.2. Temporal XML Documents

An XML document tree contains four kinds of nodes, which are root, element, attribute and text nodes, respectively. Actually, a root node is a special element node, which represents the beginning of an XML document uniquely. Connecting nodes forms the edges of an XML tree.

A temporal XML document is also represented as an XML tree. As shown in Figure 5, we attach bi-temporal annotations to tree nodes rather than to tree edges. Note that the valid or transaction time of a child node is contained by its parent node.

Definition 3. (Temporal XML document model) A temporal XML document model is a tuple \( (X, T, M) \) \([19]\), where

1. \( X = \{X_1, \ldots, X_n\} \) is a set of XML documents and an XML document (say \( X_i \in X \)) is tuple \((N_i, ED_i, \mu_i)\). Here \( N_i \) is a set of nodes (including all types of nodes in XML tree); \( ED_i = \{(n_1, n_2)\} \) \( (n_1, n_2 \in N_i) \) is a set of edges; \( \mu_i \rightarrow \epsilon \) \( E_i \cup A_i \) maps the nodes of the XML document model to an element set \( E_i \) and an attribute set \( A_i \) of an XML Schema.

2. \( T \) is a set of times.

3. \( M: X \rightarrow 2^T \) is a timestamp function that maps an XML document to a timestamp (a set of times).

Definition. (snapshot of temporal XML document) Let \( TX = (X, T, M) \) be a temporal XML document model. For \( t \in T \), snapshot \( (t, TX) = X_t \) in which \( X_t \) is \( X \) and \( t \in T(X_t) \).

An element of temporal XML documents is assigned with four special attributes vStart, vEnd, tStart and tEnd, which represent the valid and transaction interval of this element, respectively. When the attribute or text of an element is varying with time, we denote the element using a special attribute isAttr and represent the attribute or text with a subelement, in which the text node is denoted by an element named TextNode.

Let us look at an example. Suppose that we have a temporal element named phone, which has a text node which changes with time. Then we have a fragment of temporal XML document shown in Figure 6.

3.3. Temporal RDF(S)

RDF can be used to express propositions using precise formal vocabularies. W3C has given the semantics of RDF with model theory, which is regarded as "interpretation theory" in \([27]\). To represent time information in RDF, we need to add temporal information to this theory. In this paper, we adopt the temporal RDF model proposed in \([22]\) with both valid and transaction dimensions by labeling.

Definition 4. (Temporal RDF model) A simple temporal interpretation of RDF is a tuple \((I, T, M)\), in which

---

**Figure 2.** temporalAttr structure.

**Figure 3.** temporalAttrType structure.

**Figure 4.** temporalText structure.

**Figure 5.** An example of temporal XML tree.

**Figure 6.** A fragment of temporal XML document.
1. \( I = \{ I_1, \ldots, I_n \} \) is a set of simple interpretations. In \( I = (C_p, P_r, R_t, Ext, Ext, CEx) \), \( I_n \) is a set of classes; \( P_r \) is a set of properties; \( R_t \) is a set of all resources, which is actually the universe of RDF(S), containing a distinguished subset \( L \) called literal values; \( Ext: P_r \to R_t \times R_t \) is used to express the relationship between resources; \( CEx: C_t \to 2^C \) maps a class \( c \in C \) to a subset of \( C \) (i.e., \( C_t = Ext(\text{rdfs:Class}) \)), which means each element of \( C_t \) is an extension of \text{rdfs:Class}.

2. \( T \) is a set of times.

3. \( M: I \to 2^T \) is a timestamp function that maps an interpretation to a timestamp (a set of times).

**Definition 5.** (Snapshot of temporal RDF model). Let \( T_I = (I, T, M) \) be a simple interpretation. Then for \( t \in T_I \), snapshot \( s(t, T_I) = I_t \), where \( I_t \in I, t \in M(I_t) \).

A temporal triple is an RDF triple with temporal labels and the notation, denoted \( (\text{v}, \text{b}, \text{c}) \). Here \( (\text{v}, \text{b}, \text{c}) \) denotes a triple with subject \( \text{a} \), property \( \text{b} \), and object \( \text{c} \), \( \text{v} \) denotes the valid time interval, and \( \text{b} \) denotes the transaction time. Actually, the expression \( (\text{v}, \text{b}, \text{c}) \) \((\text{v}, \text{b}, \text{c}) = (\text{v}_1, \text{v}_2, \text{b}, \text{c}) \) is a notation for \( (\text{a}, \text{b}, \text{c}) \in \mathbb{V} \) \((\text{v}_1, \text{v}_2, \text{b}, \text{c}) = (\text{v}_1, \text{v}_2, \text{b}, \text{c}) \). Note that the temporal labels are applied at the level of triples, not the level of the subject, property or object of triples.

We use the classical RDF graph with temporal vocabularies to represent temporal RDF. A temporal RDF graph is a set of temporal triples, which consists of a set of \( (\text{a}, \text{b}, \text{c}) \), \( (\text{X}, \text{tsubj}, \text{a}) \), \((\text{X}, \text{tpred}, \text{b}) \), \((\text{X}, \text{tobj}, \text{c}) \), and \((\text{X}, \text{valid}, \text{v}) \), \((\text{X}, \text{trans}, \text{t}) \). Here \( X \) denotes the statement of triple \((\text{a}, \text{b}, \text{c}) \), \( \text{f} \) and \( \text{g} \) represent the valid and transaction time, respectively.

Figure 7 shows an example of temporal RDF graph.

### 4. Mapping Temporal XML to Temporal RDF(S)

As we know, XML Schema defines all elements and attributes with relevant structure and types of XML documents, which generally contain rich semantic information. So, transforming temporal XML to temporal RDF(S) is conducted at two levels. The first level is to extract temporal RDF Schema from temporal XML Schema and the second level is to transform temporal XML document to temporal RDF triples.

In this section, we propose the rules for mapping temporal XML to RDF(S) based on the formal definitions of two temporal data models.

#### 4.1. Temporal XML Schema Mapping

Some generic mapping rules from non-temporal XML Schema to RDF(S) have been proposed. But they cannot be directly applied to extract temporal RDF(S) from temporal XML Schema. For our purpose, we define a mapping function \( f: \text{XML Schema} \to \text{RDF(S)} \), which maps a snapshot \( S_t = (E_s, A_s, D, p, k, t) \) of temporal XML Schema at time point \( t \) to the corresponding temporal RDF Schema \( I_t = (C_p, P_r, R_t, Ext, CEx) \), that is,

\[
(f(S_t)) = f((E_s, A_s, D, p, k, t)) = (C_p, P_r, R_t, Ext, CEx) = I_t.
\]

The common properties of RDF Schema are presented as follows:

#### rdf:type, rdf:domain, rdf:range

Based on the formal definitions of temporal XML Schema and temporal RDF(S) above, we propose the following mapping rules, which establish the major correspondences of different components in these two temporal data models.

1. For attribute \( @a \in A_s \), we have \( f(@a) \in P_r \).

2. Simple datatypes \( D \) of XML Schema are mapped to resources of RDF. We have \( d \in D, f(d) = \text{Datatype} \in \text{Ext(rdfs: type)} \).

3. For attribute \( @a \in A_s \) and \( \kappa(@a) = dt, dt \in D \), we have \( f(\kappa(@a), dt) \in \text{Ext(2d: range)} \).

4. For element \( e \in E_s \) and \( \kappa(e) = dt, dt \in D \), we have \( f(e) \in P_r, f(e, dt) \in \text{Ext(rdfs: range)} \).

5. For element \( e \in E_s \) and \( r(e) \neq c \), we have \( f(e) \in C_t \).

Temporal information in XML Schema is directly mapped to the corresponding temporal RDF(S). To ensure the correctness of RDF(S), snapshot mapping based on time points is applied with the function \( f \). As we know, XML Schema is made up of three main building blocks: elements, attributes and type definitions and it contains many built-in datatypes (e.g., \text{xs:string} and \text{xs:decimal}). First, RDF(S) do not have built-in datatypes. Rule 2 can map the built-in datatypes of XML Schema to the instances of \text{rdfs:Datatype}, which is a predefined class of RDF(S). Second, the attributes of XML Schema are usually used to describe the features of elements which are similar to the properties of RDF(S). We can interpret the attributes of XML Schema as the properties of RDF(S) with Rule 1 and get the range information with Rule 3. Note that, however, the domain of a property is related to the attribute's location in the XML Schema, which can be extracted by specific processing. Third, we identify two categories of elements: simple elements and complex elements. The simple elements of XML Schema contain only the text without any attributes and subelements and can be directly interpreted as the properties of RDF(S), which range restrictions are obtained with Rule 4. The complex elements of XML Schema, however, are translated into the classes of RDF(S) with Rule 5, because they possess complex data structures with attributes and subelements.

Based on the mapping rules above, we present the following algorithm to map a temporal XML Schema model into a temporal RDF(S) model. For a snapshot of temporal XML Schema, it is parsed in a depth-first order and then all elements, attributes and types are mapped with the mapping rules above. Note that two declaration types \text{global} and \text{local} in XML Schema cannot be mapped directly by the formal definitions. So, the algorithm proposed above does not deal with the references themselves. The references of \text{complexType} can be mapped to classes and the elements which refer to the \text{complexType} are further mapped to subclasses of the mapped classes. In addition, we can exploit some domain information of properties by using positional relationship between elements and attributes.

#### 4.2. Temporal XML Document Mapping

With the RDF Schema mapped from the temporal XML Schema, we can map the corresponding temporal XML document to temporal RDF. We define a mapping function \( g: \text{XML document} \to \text{RDF} \), which maps a snapshot \( S_t = (N_t, ED_t, \mu) \) of temporal XML document at time point \( t \) to the corresponding RDF triple \( I_t = (C_p, P_r, R_t, Ext, CEx) \), that is,

\[
g(X) \mapsto f(N_t, ED_t, \mu).
\]

Based on the formal definitions of temporal XML document and temporal RDF above, we propose the following mapping rules which establish the major correspondences among different components in these two temporal data models.

1. If node \( n \in N_t \) and \( \mu(n) = e \in E_t \) and \( (e) \neq c \), we have \( g(n) \in R_t, (g(n), f(e)) \in Ext(rdfs: type) \).

2. If node \( n \in N_t \) and \( \mu(n) = e \in E_t \) and \( (e) \neq c \), for pair \( (n, p) \in ED_n \), we have \( g(p), g((n), f(p)) \in Ext(rdfs: subClassOf) \).

3. If node \( n \in N_t \) and \( \mu(n) = @a \in A_s \) for pair \( (n, p) \in ED_n \), we have \( g(p), g((n), f(p)) \in Ext(f(@a)) \), where \( I \) is a string value of node \( n \).
1. \( I = \{ I_1, \ldots, I_l \} \) is a set of simple interpretations. In \( I = (C, P, R, Ext, CExt) \in I, C \) is a set of classes; \( P \) is a set of properties; \( R \) is a set of all resources, which is actually the universe of RDF(S), containing a distinguished subset \( L \) called literal values; \( Ext : P \to R \times R \) is used to express the relationship between resources; \( CExt : C \to 2^P \) maps a class \( c \in C \) to a subset of \( P \) (i.e., \( C_e = Ext(rdfs:Class) \), which means each element of \( C_e \) is an extension of rdfs:Class).

2. \( T \) is a set of times.

3. \( M : I \to 2^{T} \) is a timestamp function that maps an interpretation to a timestamp (a set of times).

**Definition 5.** (Snapshot of temporal RDF model). Let \( TI = (I, T, M) \) be a simple interpretation. Then for \( t \in T, snapshot (t, TI) = I_t, \) where \( I_t \in I, t \in M(I_t) \).

A temporal triple is an RDF triple with temporal labels and the notation, denoted \((a, \mathit{tpred}, b)\). \( \mathit{tpred} \) denotes the temporal predicate and \( a, b \) denotes the subject and object respectively.

**Definition 6.** (Temporal RDF graph) A temporal RDF graph is a set of temporal triples, which establishes the major correspondences of different components in these two temporal data models.

1. For attribute \( @a \in A \), we have \( f(@a) \in P_D \).
2. Simple datatypes \( D \) of XML Schema are mapped to resources of RDF. We have \( dt \in D, (f(dt), \mathit{rdfs:Datatype}) \in Ext(rdfs: type) \).
3. For attribute \( @a \in A \), and \( k = (a) \in dt, dt \in D, \) we have \( (f(@a), f(dt)) \in Ext(rdfs: range) \).
4. For element \( e \in E, k = (e) \in dt, dt \in D, \) we have \( f(e) \in P_r, (f(e), f(dt)) \in Ext(rdfs: range) \).
5. For element \( e \in E \) and \( \tau(e) \neq e \), we have \( f(e) \in C \).

Temporal information in XML Schema is directly mapped to the corresponding temporal RDFs. To ensure the correctness of RDFs, snapshot mapping based on time points is applied with the function \( f \). As we know, XML Schema is made up of three main building blocks: elements, attributes and type definitions and it contains many built-in datatypes (e.g., xs: string and xs: decimal). First, RDFs do not have built-in datatypes. Rule 2 can map the built-in datatypes of XML Schema to the instances of rdfs: Datatype, which is a predefined class of RDFs. Second, the attributes of XML Schema are usually used to describe the features of elements which are similar to the properties of RDFs. We can interpret the attributes of XML Schema as the properties of RDFS with Rule 1 and get the range information with Rule 3. Note that, however, a domain of a property is related to the attribute's location in the XML Schema, which can be extracted by specific processing. Third, we identify two categories of elements: simple elements and complex elements. The simple elements of XML Schema contain only the text without any attributes and subelements and can be directly interpreted as the properties of RDFs, which range restrictions are obtained with Rule 4. The complex elements of XML Schema, however, are translated into the classes of RDFs with Rule 5, because they possess complex data structures with attributes and subelements.

Based on the mapping rules above, we present the following algorithm to map a temporal XML Schema model into a temporal RDFS model.

For a snapshot of temporal XML Schema, it is parsed in a depth-first order and then all elements, attributes and types are mapped with the mapping rules above. Note that two declaration types global and local in XML Schema cannot be mapped directly by the formal definitions. So, the algorithm proposed above does not deal with the references themselves. The references of complexTypes can be mapped to classes and the elements which refer to the complexTypes are further mapped to subclasses of the mapped classes. In addition, we can exploit some domain information of properties by using positional relationship between elements and attributes.

4. Mapping Temporal XML to Temporal RDF(S)

As we know, XML Schema defines all elements and attributes with relevant structure and types of XML documents, which generally contain rich semantic information. So, transforming temporal XML to temporal RDF(S) is conducted at two levels. The first level is to extract temporal RDF Schema from temporal XML Schema and the second level is to transform temporal XML document to temporal RDF triples. In this section, we propose the rules for mapping temporal XML to RDF(S) based on the formal definitions of two temporal data models. The detailed algorithm for mapping temporal XML document to RDF triples will be clarified. Not at last, we employ an example to illustrate the complete temporal processing algorithm.

4.1. Temporal XML Schema Mapping

Some generic mapping rules from non-temporal XML Schema to RDF(S) have been proposed. But they cannot be directly applied to extract temporal RDFS from temporal XML Schema. For our purpose, we define a mapping function \( f : XML \text{ Schema} \to RDFS \), which maps a snapshot \( S_t = (E, A, D, p, k, \tau) \) of temporal XML Schema at time point \( t \) to the corresponding temporal RDF Schema \( I_t = (C_p, P_r, Ext, CExt) \), that is,

\[
f(S_t) = f(E, A, D, p, k, \tau) = (C_p, P_r, Ext, CExt) = I_t.
\]

The common properties of RDF Schema are presented as follows:

- rdf:type, rdfs:domain, rdfs:range,
- rdfs: subclassOf, rdfs: subPropertyOf.

Based on the formal definitions of temporal XML Schema and temporal RDF(S) above, we propose the following mapping rules, which establish the major correspondences of different components in these two temporal data models.

1. If node \( n \in N_t \) and \( \mu(n) = e \in E, \) and \( \tau(e) \neq e \), we have \( g(n) \in R_p, (g(n), f(e)) \in Ext(rdfs: type) \).
2. If node \( n \in N_t \) and \( \mu(n) = e \in E, \) and \( \tau(e) \neq e \), for pair \( (p, n) \in ED_p, \) we have \( (g(p), g(n)) \in Ext(f(e)) \) and \( (g(n), f(l)) \in Ext(f(e)) \) where \( l \) is a string value of node \( n \).
4. If node $n \in N$, and $\mu(n) = e$ (it is a text node), there are two possibilities for pair $(p, n) \in ED$: First, if $\mu[p] = e \in E$, and $\nu(p) \neq e$, we have $(g(p), g(l)) \in Ext(rdfrd有价值)$, where $l$ is the string value of node $n$. Second, if $\mu(p) = e \in E$, and $\nu(q) \neq e$, we have $(g(\text{ancestor}), g(l)) \in Ext(f(o))$, where ancestor is the parent of node $p$.

The temporal information of XML document can be directly mapped to the corresponding temporal RDF. We map the snapshot of temporal XML document with the function $g$. As we know, XML Schema is used to constrain the structure of XML documents. All elements and attributes in XML documents come from the corresponding XML Schema. The elements of XML document are the instances of the elements of XML Schema. As mentioned in Subsection 4.1, the elements of XML Schema may be parsed into classes or properties of the RDF Schema. For the former elements of XML Schema, the corresponding elements of XML document are mapped into the instances of the mapped classes with Rule 1. For the latter elements of XML Schema, the corresponding elements of XML document, which have concrete values (say, $l$), are mapped to the predicates of objects in RDF Schema. In addition, we introduce new RDF Schema properties rdfx:describes and rdfx:hasClass to deal with two special cases. Here rdfx:describes connects the resource of the XML document itself (its URI) with the root class it describes, and rdfx:hasClass is defined to solve a problem when two directly connected labels in the XML tree are both identified as classes. It can be observed from Algorithm 2 that two time dimensions in the temporal XML document are mapped to attributes of elements. The time that limits the relationship between two resources in RDF is presented as a label. So, we use the intersection of time intervals in resources to label the relevant triples.

Note that XML data is not always time varying and some data lack temporal information. To handle this problem, we propose three additional rules in order to apply the proposed algorithm.

1. If an element (including non-temporal attributes) lacks valid information, its temporal information follows the time of the schema.
2. The valid time of an element must be contained by the valid time of its ancestors.
3. If a transaction time is absent, the algorithm follows the transaction time of the parents.

4.3. An Example Illustration

To illustrate the proposed approach in translating temporal XML into temporal RDF(S), we present an example of temporal XML about e-bookstore. The temporal XML Schema of the example is presented in Figure 8. According to the schema mapping rules and the corresponding mapping algorithm, the temporal XML Schema shown in Figure 8 is mapped to the classes and properties of temporal RDF(S), which are shown in Table 1.

```

Input: temporal XML Schema
Output: temporal RDF(S)
for each element SchemaVersion of the root element TemporalSchema {
    schemaPath = getSchemaVersionPath();
    (sv,tst) = getAttributeTimeVT();
    Map understandableXMLNode = createEmptyNodesPath();
    for each child of SchemaVersion {
        xpath = getAttributePath();
        (V,T) = getAttributeTimeVT();
        insertChildNodesMap(childNodesMap, xpath, (V,T));
    }
    parseSchema(schemaPath, childNodesMap, (sv,tst));
    parseSchema(path, map, (sv,tst))
    root = getSchemaRootElement(path);
    activeTime(AV, AT) = (sv,ST);
    visitNode(node, activeTime(AV, AT)) {
        qname = getAttributeName();
        nodeName = getAttributeName();
        xpath = getNodeXPath();
        (V,T) = check(xpath,map);
        if (V,T) = null {
            set activeTime=(AV, V, \cap (AT, T));
            if checkIfExistInRDF(nodeName) = true {
                (odV, oldT) = getExistTime(nodeName);
                setNewTime(nodeName, (V,T), (odV, oldT));
            } else {
                switch (qname) { case "element":
                    if getAttributeRef() = null {
                        if sComplexType(getAttributeRef()) = true {
                            createClass(nodeName, activeTime);
                            setParentClass(nodeName, getAttributeRef());
                            break;
                        } /*checked with mapping rules and map element to RDFS*/
                        if checkIfClass(nodeName) = true {
                            /*activeClass is a global variable to record current valid class*/
                            setActiveClass = nodeName;
                        }
                        else {
                            setPropertyDomain(nodeName, activeClass);
                            break;
                        }
                    } /*attribute*/
                    if getAttributeRef() = null {
                        break;
                    } /*checked with mapping rules and map attribute to RDFS*/
                    if sGlobal(xpath) = false {
                        setPropertyDomain(nodeName, activeClass);
                        break;
                    } /*complexType*/
                    if sGlobal(xpath) = true & onlyTemporalAttribute(xpath) = false {
                        createClass(nodeName, activeTime);
                        break;
                    } }
            }
            for each child of this node {
                visitNode(childNode, activeTime);
            }
        }
    }
}
```
4. If node $n \in N$, and $\mu(n) = \epsilon$ (it is a text node), there are two possibilities for pair $(p, n) \in ED$: First, if $\mu(p) = e \in E$, and $\epsilon \neq \epsilon$, we have $(g(p), g(l)) \in \text{Ext}(\text{rdf: value})$, where $l$ is the string value of node $n$. Second, if $\mu(p) = e \in E$, and $\epsilon \neq \epsilon$, we have $(g(\text{ancestor}), g(l)) \in \text{Ext}(\text{rdf: value})$, where $\text{ancestor}$ is the parent of node $p$.

The temporal information of XML document can be directly mapped to the corresponding temporal RDF. We map the snapshot of temporal XML document with the function $g$. As we know, XML Schema is used to constrain the structure of XML documents. All elements and attributes in XML documents come from the corresponding XML Schema. The elements of XML document are the instances of the elements of XML Schema. As mentioned in Subsection 4.1, the elements of XML Schema may be parsed into classes or properties of the RDF Schema. For the former elements of XML Schema, the corresponding elements of XML document are mapped into the instances of the mapped classes with Rule 1. For the latter elements of XML Schema, the corresponding elements of XML document, which have concrete values (say, $l$), are mapped to the predicates of specific triples with object $l$. Rule 2 and Rule 3 are used to clarify the mappings of these elements and attributes. In an XML tree, there is a kind of nodes named text nodes, which are presented as formatted strings. In Rule 4, two situations in the parent nodes (say, $p$) of the text nodes are discussed. If $p$ is converted to an instance of class, the text node is mapped to the object of the triple with subject $p$ and predicate rdf: value. If $p$ is parsed into a property, the text node is mapped to an object of the triple whose subject is the ancestor (the parent of $p$) and predicate is the property that $p$ is mapped to.

To map a temporal XML document to a temporal RDF, the temporal XML document tree is parsed in a depth-first order. Each node in the XML document tree is identified if it is a node for a property or a class. The validity of nodes is verified via temporal information according to the generated schema. In addition, the resources described in RDF have unique identifiers, denoted by URIs. So, we jointly utilize the namespaces of the temporal XML document and its location in the temporal XML document, which is presented by XPath, to represent resources while generating the instances of RDF classes. Based on the mapping rules above, we present the following algorithm to map a temporal XML document model into a temporal RDF model.

In Algorithm 2, we use resource to represent the current XML tree node and semi-space-qualifier "onto:" to denote the objects in RDF Schema. In addition, we introduce new RDF Schema properties rdfx: describes and rdfx: hasClass to deal with two special cases. Here rdfx: describes connects the resource of the XML document itself (its URI) with the root class it describes, and rdfx: hasClass is defined to solve a problem when two directly connected labels in the XML tree are both identified as classes. It can be observed from Algorithm 2 that time dimensions in the temporal XML document are mapped to attributes of elements.

Note that XML data is not always time varying and some data lack temporal information. To handle this problem, we propose three additional rules in order to apply the proposed algorithm.

1. If an element (including non-temporal attributes) lacks valid information, its temporal information follows the time of the schema.
2. The valid time of an element must be contained by the valid time of its ancestors.
3. If a transaction time is absent, the algorithm follows the transaction time of the parents.

4.3. An Example Illustration

To illustrate the proposed approach in translating temporal XML into temporal RDF(S), we present an example of temporal XML about e-bookstore. The temporal XML Schema of the example is presented in Figure 8.

According to the schema mapping rules and the corresponding mapping algorithm, the temporal XML Schema shown in Figure 8 is mapped to the classes and properties of temporal RDF(S), which are shown in Table 1.
Algorithm 2: Interpreting temporal XML document to temporal RDF.

```
Input: doc_Uri, the uri of temporal XML document
Output: temporal RDF triples
root = getRootElement(doc_Uri);
activeRes = getRootXpath();
set activeTime(VF, AT) = (infinite, (doc:CreationTime, UC));
visitNode(root, activeTime);
visitNode(node, active_time(VF, AT)) { node_name = getNodeName(); resource = getNodeXpath();
    (VF, T) = getAttributeTimeVT();
    if (VF, T) == null { 
        set (VF, T) = (infinite, infinite) 
        (XV, AT) = getSchemaTime(nodeName);
        switch(getNodeType) { 
            case "class": 
                if isfinished(getLastTriple()) == false { 
                    (LV, LT) = getTime(getLastTriple()); 
                    finishTriple(getLastTriple(), resource, (VF, XF, V), (AT, T)); 
                } else { 
                    createTriple(activeRes, rdfx:hasClass, resource, (VF, XF, V), (AT, T)); 
                } 
                onto:Class = getClass(nodeName);
                createTriple(resource, rdf:type, onto:Class, (VF, XF, V), (AT, T));
            break;
            case "property": 
                if isfinished(getLastTriple()) == false { 
                    (LV, LT) = getTime(getLastTriple()); 
                    finishTriple(getLastTriple(), activeRes, (VF, XF, V), (AT, T)); 
                } else { 
                    createTriple(activeRes, rdf:type, onto:Prop, null, (VF, XF, V), (AT, T)); 
                } 
                default: 
                    text = getText();
                    if isfinished(getLastTriple()) == false { 
                        (LV, LT) = getTime(getLastTriple()); 
                        finishTriple(getLastTriple(), text, (VF, XF, V), (AT, T)); 
                    } else { 
                        createTriple(activeRes, rdf:value, text, (VF, XF, V), (AT, T)); 
                    } 
        } 
        for each child of this node { 
            visitNode(childNode, activeTime); 
        } 
    } 
}
```

Figure 9 presents a temporal XML document which has the temporal XML Schema shown in Figure 8. Using the document mapping rules and the corresponding mapping algorithm, the temporal XML document shown in Figure 8 is mapped to the temporal RDF triples shown in Table 2, which have the temporal RDFS shown in Table 1.
Algorithm 2. Interpreting temporal XML document to temporal RDF.

Input: doc_Uri, the uri of temporal XML document

Output: temporal RDF triples

1. root = getRootElement(doc_Uri);
2. activeRes = getNodeXPath();
3. set activeTime(AV, AT) = (infinite, (doc_CreationTime, UC));
4. visitNode(root, activeTime);
5. visitNode(node, activeTime(AV, AT)) {
   nodeName = getNodeName();
   resource = getNodeXPath();
   (F, T) = getAttributeTimeVT();
   if (F, T) == null {
      set (F, T) = (infinite, infinite)
      (XY, AT) = getSchemaTime(nodeName);
      switch getNodeType {
         case "class":
            createTriple(activeRes, onto: Class = getClass(nodeName);
            if isfinished(getLastTriple()) == false {
               AV = createTriple(activeRes, rdf:desc, resource, ((AV, XY, F), (AT, T)));
            }
            else {
               finishTriple(getLastTriple(), resource, ((AV, XY, F), (AT, T)));
            }
            break;
         case "property":
            if isfinished(getLastTriple()) == false {
               (LV, LT) = getTime(getLastTriple());
               finishTriple(getLastTriple(), activeRes, ((LV, AT), (LT, T)));
            }
            else {
               createTriple(activeRes, onto: Prop = getProperty(nodeName);
            }
            break;
         default:
            text = getText();
            if isfinished(getLastTriple()) == false {
               (LV, LT) = getTime(getLastTriple());
               finishTriple(getLastTriple(), text, ((LV, AT), (LT, T)));
            }
            else {
               createTriple(activeRes, rdf: value, text, ((AV, XY, F), (AT, T)));
            }
      }
      for each child of this node {
         visitNode(childNode, activeTime);
      }
   }
}
Table 1. An example of classes and properties in temporal RDFS.

<table>
<thead>
<tr>
<th>property</th>
<th>domain</th>
<th>range</th>
<th>validTime</th>
<th>transactionTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>bookStore</td>
<td>xs:string</td>
<td>(2010-7-18, now)</td>
<td>(2010-08-01, UC)</td>
</tr>
<tr>
<td>name</td>
<td>book</td>
<td>xs:string</td>
<td>(2010-7-18, now)</td>
<td>(2010-08-01, UC)</td>
</tr>
<tr>
<td>isbn</td>
<td>book</td>
<td>xs:string</td>
<td>(2010-7-18, now)</td>
<td>(2010-08-01, UC)</td>
</tr>
</tbody>
</table>

Table 2. An example of temporal XML.

<table>
<thead>
<tr>
<th>sub</th>
<th>pred</th>
<th>obj</th>
<th>validTime</th>
<th>transactionTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>/BookStore.xml</td>
<td>rdfx:describes</td>
<td>/bookStore</td>
<td>(2010-7-18, now)</td>
<td>(2010-03-11, UC)</td>
</tr>
<tr>
<td>/BookStore</td>
<td>rdf:type</td>
<td>onto/bookStore</td>
<td>(2010-7-18, now)</td>
<td>(2010-03-11, UC)</td>
</tr>
<tr>
<td>/BookStore</td>
<td>rdf:x:value</td>
<td>This bookstore supplies...</td>
<td>(2010-7-18, now)</td>
<td>(2010-03-11, UC)</td>
</tr>
<tr>
<td>/BookStore</td>
<td>onto:owner</td>
<td>July</td>
<td>(2010-7-18, now)</td>
<td>(2010-03-11, UC)</td>
</tr>
</tbody>
</table>

5. Prototype Implementation

Based on the mapping rules and algorithms proposed above, we design and implement a prototype system called TX-TR as proof-of-concept, which can map temporal XML Schema and documents into temporal RDF(S). In the following, we briefly discuss the design and implementation of the prototype system.

The core of TX-TR is to automatically map temporal XML into temporal RDF(S). Before the mapping is carried out, TX-TR needs to read in temporal XML Schema as its input, which is used to extract temporal RDF(S). Then, temporal XML document needs to be read in and mapped, which is done to obtain temporal RDF triples. TX-TR displays temporal RDF triples as its output after the mapping is completed. The overall architecture of TX-TR is shown in Figure 10, which includes three main modules, namely, Parse module, Mapping module and Interface module.

1. Parse module. This module parses inputs of temporal XML Schema, temporal XML document and stores the parsed schema information. Schema information (e.g., file Path, temporal information, element constraints) of XML Schema can be extracted and represented.

2. Mapping module. This module transforms the parsed results into the corresponding temporal RDF(S) by using the mapping rules and algorithms proposed above.

3. Interface module. This module provides an interface to users, which mainly produces and finally displays the resulting RDF.

Implementation of TX-TR is based on Java with the API of dom4j and XPath over Eclipse platform, and the Graphical User Interface (GUI) is exploited over eclipse platform. TX-TR is implemented and run with a PC (CPU 2.5GHz, RAM 8GB and Windows 10 system).

The screen snapshot of TX-TR running one of the case studies is shown in Figure 11 and Figure 12. It can be seen from Figure 11 and Figure 12 that the graphical user interface of TX-TR contains two parts: "Temporal Schema → RDFS" and "Temporal XML document → RDF". Figure 10 shows the first part that maps temporal XML Schema into temporal RDFS, in which "Location" is used to import a file of temporal XML Schema and "Export RDFS" is used to export the corresponding temporal RDF transformed the imported temporal XML document.

6. Conclusions

In this paper, we present how to map temporal XML document with Schema into temporal RDF(S). The mapping is conducted at two levels. The first level of mapping is to map the temporal XML Schema into temporal RDF Schema and the second level of mapping is to map the temporal XML document into temporal RDF triples. For each level of mapping, we
5. Prototype Implementation

Based on the mapping rules and algorithms proposed above, we design and implement a prototype system called TX-TR as proof-of-concept, which can map temporal XML Schema and documents into temporal RDF(S). In the following, we briefly discuss the design and implementation of the prototype system.

The core of TX-TR is to automatically map temporal XML into temporal RDF(S). Before the mapping is carried out, TX-TR needs to read in temporal XML Schema as its input, which is used to extract temporal RDF(S). Then, temporal XML document needs to be read in and mapped, which is done to obtain temporal RDF triples. TX-TR displays temporal RDF triples as its output after the mapping is completed. The overall architecture of TX-TR is shown in Figure 10, which includes three main modules, namely, Parse module, Mapping module and Interface module.

1. Parse module. This module parses inputs of temporal XML Schema, temporal XML document and stores the parsed schema information. Schema information (e.g., file Path, temporal information, element constraints) of XML Schema can be extracted and represented.

2. Mapping module. This module transforms the parsed results into the corresponding temporal RDF(S) by using the mapping rules and algorithms proposed above.

3. Interface module. This module provides an interface to users, which mainly produces and finally displays the resulting RDF.

Implementation of TX-TR is based on Java with the API of dom4j and XPath over Eclipse platform, and the Graphical User Interface (GUI) is exploited over eclipse platform. TX-TR is implemented and run with a PC (CPU 2.5GHz, RAM 8GB and Windows 10 system).

The screen snapshot of TX-TR running one of the case studies is shown in Figure 11 and Figure 12. It can be seen from Figure 11 and Figure 12 that the graphical user interface of TX-TR contains two parts: "Temporal Schema → RDFS" and "Temporal XML document → RDF". Figure 10 shows the first part that maps temporal XML Schema into temporal RDF(S), in which "Location" is used to import a file of temporal XML Schema and "Export RDFS" is used to export the corresponding temporal RDF transformed the imported temporal XML document.

6. Conclusions

In this paper, we present how to map temporal XML document with Schema into temporal RDF(S). The mapping is conducted at two levels. The first level of mapping is to map the temporal XML Schema into temporal RDF Schema and the second level of mapping is to map the temporal XML document into temporal RDF triples. For each level of mapping, we
propose the mapping rules and mapping algorithm. We illustrate our mapping approach with an example and implement a prototype system. In our mapping approach, we apply the interval operations for time calculation. We also apply snapshots to describe the relation between theoretical models. Note that obtaining a snapshot is time-consuming. So, we propose some rules and constraints to ensure the effectiveness of extracting time information. In addition, in our mapping approach, we fully consider the semantics of the temporal XML and try to preserve the semantics in the mapping as much as possible. However, the cardinality constraint in XML cannot be directly represented by RDF(S). Also, the order indicators in XML Schema are mapped into the containers or collections of RDF in literature, but the containers and collections are only used to describe instances and cannot describe the inclusion relationship between classes of RDFS. We will dedicate ourselves to deal with their mapping in our future work.

Acknowledgement

This work was supported in part by National Natural Science Foundation of China (61772269).

References


propose the mapping rules and mapping algorithm. We illustrate our mapping approach with an example and implement a prototype system. In our mapping approach, we apply the interval operations for time calculation. We also apply snapshots to describe the relation between theoretical models. Note that obtaining a snapshot is time-consuming. So, we propose some rules and constraints to ensure the effectiveness of extracting time information. In addition, in our mapping approach, we fully consider the semantics of the temporal XML and try to preserve the semantics in the mapping as much as possible. However, the cardinality constraint in XML cannot be directly represented by RDFS(S). Also, the order indicators in XML Schema are mapped into the containers or collections of RDF in literature, but the containers and collections are only used to describe instances and cannot describe the inclusion relationship between classes of RDFS. We will dedicate ourselves to deal with their mapping in our future work.

Acknowledgement

This work was supported in part by National Natural Science Foundation of China (61772269).

References