Temporal Ontology Language for Representing and Reasoning Interval-Based Temporal Knowledge

Sang-Kyun Kim[1], Mi-Young Song[1], Chul Kim[1], Sang-Jun Yea[1], Hyun Chul Jang[1], and Kyu-Chul Lee[2]

[1] Korea Institute of Oriental Medicine, South Korea
{skkim,smyoung,chulnice,tomita,hcjang}@kiom.re.kr
[2] Dept. of Computer Engineering, Chungnam National University, South Korea
kclee@cnu.ac.kr

Abstract. W3C Web Ontology working group has recently developed OWL as an ontology language for the Semantic Web. However, because OWL does not have the full-fledged semantics for temporal information, it cannot perform reasoning about temporal knowledge. Entities in the real world are changing according to the passage of time and new facts are occurring due to events. If knowledge in the KBs does not have the temporal information, it becomes incomplete and incorrect. Therefore, we in this paper propose an ontology language TL-OWL, which extends OWL to have the temporal semantics in order to represent and reason the temporal information in the Semantic Web.

1 Introduction

Semantic Web has a vision that a machine understands and processes information in web automatically by describing semantics to web. For a machine to process information, knowledge that a machine and humans can share must be described. Semantic Web provides knowledge on web resources by using ontology. OWL is an ontology language for the Semantic Web that has recently been developed by the W3C Web Ontology Working Group. However, since OWL does not have time information, questions depending on time cannot be accurately processed.

For example, let's assume as follows: When four cases happen, each case has time interval x, y, u, v according to when it happens and the relation between time interval is x before y, y overlaps u, and u before v. OWL describes each four cases and time as individuals, and the time relationship can be connected by property of the individuals. However, since OWL cannot perform the transitive reasoning among time relations, the relation of x before v cannot be reasoned. Instead, if the rule-based reasoning such as before(x,v) :- before(x,y) & overlaps(y,u) & before(u,v), questions can be answered. However, generally, the problem of the rule-based reasoning has known to be the semi-decidable. But if semantics on time to ontology can be provided, the temporal reasoning based on ontology such as x before y $\vee$ y overlaps u $\vee$ u before v $\rightarrow$ x before v is possible without using the rule-based reasoning.

J. Domingue and C. Anutariya (Eds.): ASWC 2008, LNCS 5367, pp. 31–45, 2008.

In artificial intelligent field, the researches [3] that represent and reason the temporal concepts by using Temporal Description Logics that deals with time based on Description Logics have been suggested. Researches on Temporal Description Logics are classified into Point-based Description Logics [9,12] and Interval-based Description Logics [2,4,5,10] according to how time information can be formalized. However, it is difficult to determine which way has better expression and reasoning. In order to provide the temporal reasoning in Semantics Web, the capability of OWL DL is needed, but both methods do not have that.

Therefore, in order to solve problems of OWL and Temporal Description Logics, we propose an interval-based temporal web ontology language TL-OWL which is extended language of OWL to have semantic on time interval.

The remainder of this paper is organized as following. In section 2, we briefly introduce $\mathcal{TL}$-$\mathcal{ALCF}$. In section 3, we propose an interval-based temporal web ontology language, TL-OWL. In section 4, we compare our work with prior efforts for the related subjects. Finally, in section 5, we summarize this paper.

2 A Temporal Description Logic

In this section, we briefly introduce a class of interval-based temporal Description Logic, $\mathcal{TL}$-$\mathcal{ALCF}$ proposed by Artale and Franconi. They show that the subsumption problem is decidable and supply sound and complete procedures for computing subsumption. $\mathcal{TL}$-$\mathcal{ALCF}$ is composed by the temporal logic $\mathcal{TL}$ which is able to express temporally quantified terms and the non-temporal Description Logic $\mathcal{ALCF}$ [6] extending $\mathcal{ALC}$ with features (i.e., functional roles). In this formalism an action is represented through temporal constraints on world states where each state is a collection of properties of the world holding at a certain time. The intended meaning of $\mathcal{TL}$-$\mathcal{ALCF}$ is explained as the following example.

$$\texttt{Reserve-Flight} \doteq \Diamond(x\ y)\ (\sharp\ \mathsf{f}\ x)(\sharp\ \mathsf{m}\ y).\ ((\star\texttt{TICKET} : \texttt{Unreserved})@x \sqcap (\star\texttt{TICKET} : \texttt{Reserved})@y)$$

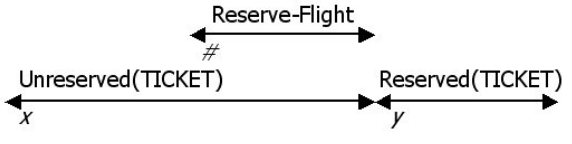

Fig. 1. Temporal dependencies of the intervals in which `Reserve-Flight` holds

Fig. 1 shows the temporal dependencies of the intervals in which the concept `Reserve-Flight` holds. `Reserve-Flight` denotes any action occurring at some interval involving a $\star$`TICKET` that was once unreserved and then reserved, where $\star$`TICKET` is a parametric feature and `Reserved` and `Unreserved` are non-temporal concepts. The parametric feature $\star$`TICKET` plays the role of formal parameter of the action, mapping any individual action of type `Reserve-Flight` to the ticket to be reserved, independently from time. Temporal variables are introduced by the temporal existential quantifier "$\Diamond$" – excluding the special temporal variable

♯, usually called now, and intended as the occurring time of the action type being defined. The temporal constraints (♯ f x)(♯ m y) state that the interval denoted by x should finish with the interval denoted by ♯ and that ♯ should meet y, where f and m are Allen's temporal relations [1] of Fig. 2.

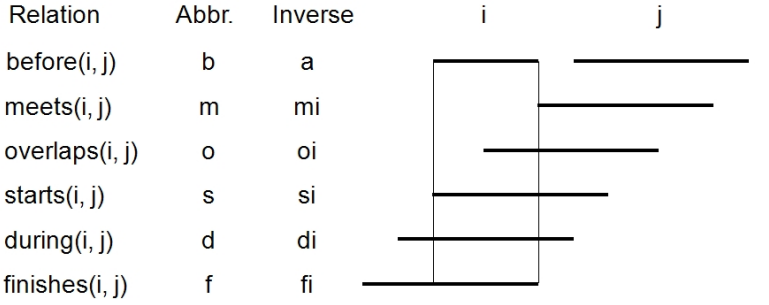

Fig. 2. The Allen's interval relationships

As the evaluation of concept at the interval, (⋆TICKET : Unreserved)@x and (⋆TICKET : Reserved)@y state that ⋆TICKET : Unreserved is qualified at x and ⋆TICKET : Reserved is qualified at y. In the concept description, the operator : is the selection of feature, which is the role quantification that is interpreted as a partial function. The following table shows the syntax of $\mathcal{TL}$-$\mathcal{ALCF}$. See [2] for the detailed semantic descriptions of $\mathcal{TL}$-$\mathcal{ALCF}$.

Table 1. The syntax of $\mathcal{TL}$-$\mathcal{ALCF}$

$\mathcal{TL}$	E, F	$\rightarrow C \mid$	(non-temporal concept)
		$E \sqcap F \mid$	(conjunction)
		$E@X \mid$	(qualifier)
		$E[Y]@X \mid$	(substitutive qualifier)
		$\Diamond(\overline{X})\overline{Tc}.E$	(existential quantifier)
	Tc	$\rightarrow$ (♯ (V) Y) | (X (V) ♯) | (X (V) Y)	(temporal constraint)
	$\overline{Tc}$	$\rightarrow Tc \mid Tc\ \overline{Tc}$	
	V, W	$\rightarrow V\ ,\ W \mid$	(disjunction)
		b | a | m | mi | o | oi |	(Allen's relations)
		s | si | d | di | f | fi | =	
	X, Y	$\rightarrow$ x | y | z | ...	(temporal variables)
	$\overline{X}$	$\rightarrow X \mid X\ \overline{X}$	
$\mathcal{ALCF}$	C, B	$\rightarrow A \mid$	(atomic concept)
		$\top \mid \bot \mid \neg C \mid C \sqcap B \mid C \sqcup B \mid \forall R.C \mid \exists R.C \mid$	
		$p \downarrow q \mid$	(agreement)
		$p \uparrow q \mid$	(disagreement)
		$p \uparrow \mid$	(undefinedness)
		$p : C \mid$	(selection)
	p, q	$\rightarrow f \mid$	(atomic feature)
		$\star g \mid$	(atomic parametric feature)
		$p \circ q \mid$	(path)

3 A Temporal Web Ontology Language

We propose an interval-based temporal ontology language TL-OWL, which adds a temporal language $\mathcal{TL}$ to OWL DL, where $\mathcal{TL}$ is the temporal part($\mathcal{TL}$) of $\mathcal{TL}$-$\mathcal{ALCF}$ as introduced in Sect. 2.

3.1 Requirements of TL-OWL

W3C Web Ontology Working Group has recently developed OWL which is an ontology language for the Semantic Web. In OWL specification, OWL is defined in two forms of syntax; First, OWL has a frame-like abstract syntax which can be easily understood and created. Second, OWL has a RDF/XML exchange syntax interpreted as RDF graphs since OWL is defined as an extension to RDF. The direct model-theoretic semantic and the RDF-compatible model-theoretic semantic are also defined to provide a formal meaning for the abstract syntax and the exchange syntax, respectively. A mapping from the abstract syntax to RDF graphs is defined and the two model semantics are shown to have the same consequences on OWL ontologies that can be written in the abstract syntax.

OWL however cannot perform the temporal reasoning since it does not have the temporal semantics as introduced in Sect. 1. The reasoning capability is usually provided in the temporal description logics [3], but all of them cannot reach the expressivity of OWL DL. Therefore, in this paper we propose *an interval-based temporal ontology language TL-OWL(TemporaL Web Ontology Language)*, which adds the temporal semantics to OWL DL.

In order to describe TL-OWL, we follow steps of the OWL specification: First, we define a high-level abstract syntax for TL-OWL. Second, we define two formal semantics for TL-OWL. One of these semantics is the direct model-theoretic semantics for TL-OWL ontologies written in the abstract syntax. The other is the RDF-compatible model-theoretic semantics as an extension of the RDF semantics, which provides semantics for TL-OWL ontologies written in the exchange syntax. And third, a mapping from the abstract syntax to RDF graphs is defined and the two model theories are shown to have the same consequences on TL-OWL ontologies. Finally, we show the reasoning in TL-OWL.

3.2 Abstract Syntax

An abstract syntax for OWL is needed since OWL is not very readable when written as RDF triples. This abstract syntax is closer to that of a frame language like OIL. As for a similar way with OWL, in this section, we define an abstract syntax for TL-OWL in the form of EBNF(Extended BNF) in Table 2. OWL syntax is not given in this table, but only OWL constructors required to understand TL-OWL are written as an italic font.

Temporal concepts in $\mathcal{TL}$-$\mathcal{ALCF}$ can be represented of the form: $\Diamond(\overline{X})(\overline{Tc})$. $(Q_0 \sqcap Q_1@X_1 \sqcap ... \sqcap Q_n@X_n)$, where $\overline{X}$ is a set of temporal variables, $\overline{Tc}$ is a set of temporal constraints, and $(Q_0 \sqcap Q_1@X_1 \sqcap ... \sqcap Q_n@X_n)$ is a conjunction of qualifiers. However, there is a problem that this normal form cannot be

Table 2. The EBNF version of an abstract syntax

Abstract Syntax
vID (variableID) ::= **URIreference** **featureID** ::= **URIreference**
axiom ::= 'TemporalClass(' **classID** ['Deprecated'] { **annotation** } **temporalDescription** ')' **axiom** ::= 'TemporalVariable(' **vID** ['Deprecated'] { **annotation** } { **temporalRelation** } ')' **temporalDescription** ::= 'intersectionOf(' **Qualification** { **Qualification** } ')' **Qualification** ::= 'Qualification(onVariable(' **vID** ') bindVariable(' **description** '))' | 'Qualification(onVariable(' **vID** ') onSubstitutiveVariable(' **vID** ') bindSubstitutiveVariable(' **description** '))' **temporalRelation** ::= 'before(' **vID** ')' | 'after(' **vID** ')' | 'meets(' **vID** ')' | 'metBy(' **vID** ')' | 'overlaps(' **vID** ')' | 'overlappedBy(' **vID** ')' | 'starts(' **vID** ')' | 'startedBy(' **vID** ')' | 'during(' **vID** ')' | 'contains(' **vID** ')' | 'finish(' **vID** ')' | 'finishedBy(' **vID** ')' | 'equal(' **vID** ')'
axiom ::= *'DatatypeProperty(' **datavaluedPropertyID** ...* ['*Functional*' | 'ParametricFunctional'] ... ')' | *'ObjectProperty(' **individualvaluedPropertyID** ...* ['*Functional*' | 'ParametricFunctional' | '*InverseFunctional*' | 'InverseParametricFunctional'] ... ['pathOf(' **featureID featureID** ')'] ')'
description ::= ***classID*** | ***restriction*** | **feature** | *'intersectionOf(' **description** ')'* | *'unionOf(' { **description** } ')'* | *'complementOf(' { **description** } ')'* | *'oneOf(' { **individualID** } ')'*
restriction ::= *'restriction(' **datavaluedPropertyID dataRestrictionComponent*** *{ **dataRestrictionComponent** } ')'* | *'restriction(' **individualvaluedPropertyID individualRestrictionComponent*** *{ **individualRestrictionComponent** } ')'* ***dataRestrictionComponent*** ::= *'allValuesFrom(' **dataRange** ')'* | *'someValuesFrom(' **dataRange** ')'* | 'selectValuesFrom(' ***dataRange*** ')' | *'value(' **dataLiteral** ')'* | ***cardinality*** ***individualRestrictionComponent*** ::= *'allValuesFrom('**description**')'* | *'someValuesFrom('**description**')'* | 'selectValuesFrom(' ***description*** ')' | *'value(' **individualID** ')'* | ***cardinality***
feature ::= 'agreementOf(' **featureID featureID** ')' | 'disagreementOf(' **featureID featureID** ')' | 'undefinednessOf('**featureID** ')'

represented as RDF triples. Therefore, in this paper we propose *an abstract syntax and a RDF/XML exchange syntax for TL-OWL*, which can be represented as RDF graphs.

TL-OWL has four axioms for TL-OWL classes and properties of TemporalClass, TemporalVariable, DatatypeProperty, and ObjectProperty. A TemporalClass can represent a temporal concept in TL-OWL. A TemporalClass contains one or more temporalDescriptions as properties and a temporalDescription contains a conjunction of Qualifications which bind a temporal variable and a non-temporal concept. A bindSubstitutiveVariable denotes a temporal substitutive qualifier which renames the variable Y to X and supplies a way of making coreference between two temporal variables. A TemporalVariable can represent the constraints among temporal variables. The TemporalVariable is identified with a variableID and has one or more of Allen's temporal relations as properties, where each temporal relation can refer another TemporalVariable. A DatatypeProperty and a ObjectProperty are the axioms defined in OWL, but in TL-OWL the ObjectProperty can have the additional types of ParametricFunctional and InverseParametricFunctional, and pathOf – a construct of the feature logic [6] – to represent a path between two featureIDs. A DatatypeProperty can have only an additional type of ParametricFunctional.

The description of OWL can contain constructs for feature logics which consist of agreementOf, disagreementOf, and undefinednessOf. A selectValuesFrom is declared within the restriction constructor for a selection operator (:).

By using above axioms and constructors, we can see that all the interval-based temporal concepts can be represented as the abstract syntax along with

preserving temporal semantics. It is easy to proof by checking the syntax of temporal concepts inductively. For an example, the `Reserve-Flight` concept of Fig. 1 can be represented to an abstract syntax as follows:

```
TemporalClass ( ex:Reserve-Flight
  intersectionOf (
    Qualification (
      onVariable ( ex:x )
      bindVariable (
        restriction (
          onProperty ( ex:TICKET )
          selectValuesFrom ( ex:Unreserved )
    )))
    Qualification (
      onVariable ( ex:y )
      bindVariable (
        restriction (
          onProperty ( ex:TICKET )
          selectValuesFrom ( ex:Reserved )
    )))))
TemporalVariable(ex:x finishedBy NOW)
TemporalVariable(ex:y metBy NOW)
```

Fig. 3. The abstract syntax of the `Reserve-Flight` concept

3.3 Direct Model-Theoretic Semantics

The direct model-theoretic semantics for TL-OWL goes directly from ontologies in the abstract syntax to a standard model theory.

Vocabularies and Interpretations. When considering a TL-OWL ontology, the vocabulary must include all the URI references and literals in that ontology. The following is a definition for a TL-OWL vocabulary.

Definition 1. *A TL-OWL vocabulary V consists of a set of literals V_L and nine sets of URI references,* $\mathrm{V_C}$, V_{TC}, V_{TV}, V_D, V_I, V_{DP}, V_{IP}, V_{AP}, *and* V_O. *In any vocabulary V_C, V_{TC}, V_{TV}, and V_D are disjoint and V_{DP}, V_{IP}, V_{AP}, and V_{OP} are pairwise disjoint. V_C, the class names of a vocabulary, contains owl:Thing and owl:Nothing. V_{TC}, the temporal class names of a vocabulary, V_{TV}, the temporal variable names of a vocabulary, V_D, the datatype names of a vocabulary, contains the URI references for the built-in OWL datatypes and rdfs:Literal. V_{AP}, the annotation property names of a vocabulary, contains owl:versionInfo, rdfs:label, rdfs:comment, rdfs:seeAlso, and rdfs:isDefinedBy. V_{IP}, the individual-valued property names of a vocabulary, V_{DP}, the data-valued property names of a vocabulary, V_{OP}, the URI references for the built-in TL-OWL ontology properties, and V_I, the individual names of a vocabulary, V_O, the ontology names of a vocabulary, do not have any required members.*

Definition 2. *A datatype map D is a partial mapping from URI references to datatypes that maps xsd:string and xsd:integer to the appropriate XML Schema datatypes.*

Definition 3. *Let D be a datatype map. An abstract TL-OWL interpretation with respect to D with vocabulary V_L, V_C, V_{TC}, V_{TV}, V_D, V_I, V_{DP}, V_{IP}, V_{AP}, V_O is a tuple of the form: $I = \langle R, T, EC, ER, L, S, LV \rangle$ where (with P being the power set operator)*

Definition. 2 is same as that of the OWL specification. Definition. 1 and Definition. 3 are similar to the definition for the OWL vocabulary and the abstract OWL interpretation except that temporal semantics are added. Thus, in this section we do not explain those of the OWL interpretations in details.

EC provides meaning for URI references that are used as TL-OWL classes and datatypes. ER provides meaning for URI references that are used as TL-OWL properties. As for the formal semantics given in $\mathcal{TL}$-$\mathcal{ALCF}$, TL-OWL classes and properties have semantics for temporal structure T such as EC: $\mathrm{V_{TC}} \rightarrow \mathrm{P(T{\times}O)}$ and ER: $\mathrm{V_{DP}} \rightarrow \mathrm{P(T{\times}O{\times}LV)}$, where O is URI references and LV is the literal values. L provides meaning for typed literals. S provides meaning for URI references that are used to denote TL-OWL individuals.

Interpretations for Constructs. EC is extended to the syntactic constructs of qualifications, descriptions, temporal relations, features as follows:

NOW, a built-in TL-OWL temporal variable, denotes the current interval of evaluation. The thirteen temporal relations defined as built-in TL-OWL properties can represent the temporal network between two temporal variables. The formal semantics for EC and ER in $\mathcal{TL}$-$\mathcal{ALCF}$ are defined as $\mathrm{EC_{V,t,H}}$ and $\mathrm{ER_{V,t,H}}$, where V is a variable assignment function associating an interval value to a temporal variable, t is an interval, and H is a set of constraints over the assignments. We in this paper omit the subscripts to simplify notations if there are not any misunderstandings. We denote the domain of partial functions by dom, which can be interpreted as a Functional type of properties.

Interpretations for Axioms. An abstract TL-OWL interpretation, I, satisfies TL-OWL axioms in the following table. Optional parts of axioms are given in square brackets ([...]). tr_i $(1 \leq \mathrm{i} \leq \mathrm{n})$ can be one out of the thirteen temporal relations in TemporalVariable.

Interpretations of Ontology. The definitions for the satisfiability, consistency, and entailment of TL-OWL ontology are given in this section. These definitions will be used in Definition. 8 and Theorem. 1.

Definition 4. *Let D be a datatype map. An Abstract TL-OWL interpretation, I, with respect to D with vocabulary consisting of V_L, V_C, V_{TC}, V_{TV}, V_D, V_I, V_{DP}, V_{IP}, V_{AP}, V_O,* **satisfies** *a TL-OWL ontology, O, iff*

1. *each URI reference in O used as a class ID (temporal class ID, temporal variable ID, datatype ID, individual ID, data-valued property ID, individual-valued property ID, annotation property ID, annotation ID, ontology ID) belongs to V_C (V_{TC}, V_{TV}, V_D, V_I, V_{DP}, V_{IP}, V_{AP}, V_O, respectively);*

2. *each literal in O belongs to V_L;*
3. *I satisfies each directive in O, except for Ontology Annotations;*
4. *there is some o ∈ R with ⟨o,S(owl:Ontology)⟩ ∈ ER(rdf:type) such that for each Ontology Annotation of the form Annotation(p v), ⟨o,S(v)⟩ ∈ ER(p) and that if O has name n, then S(n) = o; and*
5. *I satisfies each ontology mentioned in an owl:imports annotation directive of O.*

Definition 5. *A collection of abstract TL-OWL ontologies and axioms and facts is* **consistent** *with respect to datatype map D iff there is some interpretation I with respect to D such that I satisfies each ontology and axiom and fact in the collection.*

Definition 6. *A collection O of abstract TL-OWL ontologies and axioms and facts* **entails** *an abstract TL-OWL ontology or axiom or fact O' with respect to a datatype map D if each interpretation with respect to map D that satisfies each ontology and axiom and fact in O also satisfies O'.*

3.4 Mapping to RDF Graphs

We in this section provide a mapping from the abstract syntax to the exchange syntax for TL-OWL. Further, in Sect. 3.5 we show that this mapping preserves the meaning of TL-OWL ontologies.

The exchange syntax for TL-OWL is RDF/XML and the meaning of a TL-OWL ontology in RDF/XML is determined from the RDF graph that results from the RDF parsing of the RDF/XML document. The way of translating a TL-OWL ontology in abstract syntax form into the exchange syntax is by giving a transformation of each directive into a collection of RDF triples. Table reftable:Triples gives the transformation rules that transform the abstract syntax of Table 3 and Table 4 to the TL-OWL exchange syntax. The left column of the table is an abstract syntax, the center column is its transformation into triples, and the right column is an identifier for the main node of the transformation. Repeating components are listed using ellipses, as in Qualification_1 ... Qualification_n. Optional portions are enclosed in square brackets. The triples in the transformation rules that may or may not be generated are indicated by flagging with [opt]. Some transformations in the table are for directives. Other transformations are for parts of directives. Thus, the transformation rules for the directives call for other rules for components of the directive.

Table 3. Interpretations for Constructs

Abstract Syntax	Interpretations
NOW	EC(NOW) is a current interval
before(x)	$\{[u_1,v_1] \in$ T | EC(x)=$[u_2,v_2]$ implies $v_1 < u_2\}$
other temporal relations ...	...
qualification(x bindVariable(c))	EC(c), t=V(x)
qualification(x y bindSubstitutiveVariable(c))	EC(c), H=H∪{y→V(x)}
restriction(p selectValuesFrom(e))	{x ∈ domp | ER(p)(x) ∈ EC(e)}
agreementOf(p q)	{x ∈ domp ∩ domq | ER(p)(x) = ER(q)(x)}
disagreementOf(p q)	{x ∈ domp ∩ domq | ER(p)(x) ≠ ER(q)(x)}
undefinednessOf(p)	O \ domp

Table 4. Interpretations for Axioms

Abstract Syntax	Interpretations
TemporalClass(c q_1 ... q_n)	EC(c) = EC(q_1) ∩ ... ∩ EC(q_n)
TemporalVariable(c tr_1 ... tr_n)	EC(c) = ER(tr_1) ∪ ... ∪ ER(tr_n)
DatatypeProperty(p ... [ParametricFunctional])	[ER(p) is parametric functional]
ObjectProperty(p ... [ParametricFunctional] [InverseParametricFunctional] [pathOf(x y])	[ER(p) is parametric functional] [ER(p) is inverse parametric functional] [⟨u,v⟩ ∈ ER(x) ∩ ⟨v,w⟩ ∈ ER(y) implies ⟨u,w⟩ ∈ ER(p), u∈domx, v∈domy]

Table 5. Transformation to Triples

Abstract Syntax (and sequences) - S	Transformation - T(S)	Main Node - M(T(S))
vID	vID rdf:type tl:TemporalVariable .	vID
featureID	featureID rdf:type owl:FunctionalProperty . featureID rdf:type tl:ParametricFunctionalProperty [opt] .	featureID featureID
Qualification(vID C)	_:x rdf:type tl:Qualification . _:x rdf:type rdfs:Class [opt] . _:x tl:onVariable T(vID) . _:x tl:bindVariable T(C) .	_:x
Qualification(vID_1 vID_2 C)	*similar*	_:x
restriction(ID selectValuesFrom(selection))	_:x rdf:type owl:Restriction . _:x rdf:type rdfs:Class . [opt] _:x owl:onProperty T(ID) . _:x tl:selectValuesFrom T(selection) .	_:x
TemporalClass(classID [Deprecated] $annotation_1$ ... $annotation_m$ $Qualification_1$ ... $Qualification_n$	classID rdf:type tl:TemporalClass . classID rdf:type rdfs:Class [opt] . [classID rdf:type owl:DeprecatedClass .] classID T($annotation_1$) ... classID T($annotation_m$) . classID owl:intersectionOf T(SEQ $Quantification_1$ ... $Quantification_n$) .	
TemporalVariable(classID [Deprecated] $annotation_1$ ... $annotation_m$ $temporalRelation_1$ ... $temporalRelation_n$	classID rdf:type tl:TemporalVariable . classID rdf:type rdfs:Class [opt] . [classID rdf:type owl:DeprecatedClass .] classID T($annotation_1$) ... classID T($annotation_m$) . classID owl:unionOf T(SEQ $temporalRelation_1$ ... $temporalRelation_n$) .	
ObjectProperty(ID [Deprecated] $annotation_1$ ... $annotation_m$ [ParametricFunctional| InverseParametricFunctional] pathOf($featureID_1$ $featureID_2$) ...)	ID rdf:type owl:ObjectProperty . ID rdf:type rdf:Property . [opt] [ID rdf:type owl:DeprecatedProperty .] ID T($annotation_1$) ... ID T($annotation_m$) . [ID rdf:type tl:ParametircFunctional .] [ID rdf:type tl:InverseParametircFunctional .] [ID tl:pathOf T(SEQ $featureID_1$ $featureID_2$) .] ...	
DatatypeProperty(ID	*similar*	
agreementOf($featureID_1$ $featureID_2$)	_:x rdf:type rdfs:Class . _:x tl:agreementOf T(SEQ $featureID_1$ $featureID_2$) .	_:x
disagreementOf($featureID_1$	*similar*	_:x
undefinednessOf(featureID)	*similar*	_:x
before(vID_1 vID_2)	vID_1 rdf:type tl:temporalVariable . vID_2 rdf:type tl:temporalVariable . vID_1 before vID_2 .	
other temporal relations ...	...	

The `Reserve-Flight` concept in an abstract syntax form can be transformed into RDF graphs by using the above transformation rules. The following is the exchange syntax of the `Reserve-Flight` concept in the form of RDF/XML.

```
<tl:TemporalClass rdf:about="#Reserve-Flight">
  <owl:intersectionOf rdf:parseType="Collection">
```

```
    <tl:Qualification>
    <tl:onVariable rdf:resource="#x"/>
      <tl:bindVariable>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#TICKET"/>
          <tl:selectValuesFrom rdf:resource="#Unreserved"/>
        </owl:Restriction>
      </tl:bindVariable>
    </tl:Qualification>
    <tl:Qualification>
      <tl:onVariable rdf:resource="#y"/>
      <tl:bindVariable>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#TICKET"/>
          <tl:selectValuesFrom rdf:resource="#Reserved"/>
        </owl:Restriction>
      </tl:bindVariable>
    </tl:Qualification>
  </owl:intersectionOf>
</tl:TemporalClass>
<tl:TemporalVariable rdf:about="#x">
  <tl:finishedBy rdf:resource="#tl:NOW"/>
</tl:TemporalVariable>
<tl:TemporalVariable rdf:about="#y">
  <tl:metBy rdf:resource="#tl:NOW"/>
</tl:TemporalVariable>
```

Fig. 4. The RDF/XML representation of the `Reserve-Flight` concept

3.5 RDF-Compatible Model-Theoretic Semantics

The model-theoretic semantics for TL-OWL is defined as an extension of the RDF semantics. There is a correspondence between the direct model-theoretic semantics for the abstract syntax and the semantics defined in this section. As a way noted in the OWL specification, if any conflict should ever arise between these two forms, then the direct model-theoretic semantics takes precedence.

From the RDF semantics and the OWL semantics, for V a set of URI references and literals containing the RDF and RDFS vocabulary and D a datatype map, a D-interpretation of V is a tuple $I = \langle R_I, T_I, P_I, EXT_I, S_I, L_I, LV_I \rangle$. R_I is the domain of discourse. T_I is the time intervals of I, P_I is a subset of R_I, the properties of I. EXT_I is used to give meaning to properties, and is a mapping from P_I to $P(R_I \times R_I)$. S_I is a mapping from URI references in V to their denotations in R_I. L_I is a mapping from typed literals in V to their denotations in R_I. LV_I is a subset of R_I that has literal values.

The set of classes C_I is defined as $C_I = \{x \in R_I \mid \langle x, S_I(\text{rdfs:Class}) \rangle \in EXT_I (S_I(\text{rdf:type})) \wedge \langle T_I, x \rangle \in C_I\}$, and the mapping $CEXT_I$ from C_I to $P(R_I)$ is defined as $CEXT_I(c) = \{x \in R_I \mid \langle x, c \rangle \in EXT_I(S_I(\text{rdf:type})) \wedge \langle T_I, x \rangle \in C_I\}$.

Definition 7. *Let D be a datatype map that includes datatypes for rdf:XML Literal, xsd:integer and xsd:string. A TL-OWL interpretation, $I = \langle R_I, T_I, P_I, EXT_I, S_I, L_I, LV_I \rangle$, of a vocabulary V, where V includes the RDF and RDFS vocabularies, is a D-interpretation of V that satisfies all the constraints in this section.*

The following tables from Table 6 to Table 10 give the constraints of TL-OWL directives and constructs for the RDF-compatible model-theoretic semantics with the D-interpretation.

Table 6. Conditions concerning parts of TL-OWL universe and syntactic categories

If E is	then			Note
	$S_I(E) \in$	$CEXT_I(S_I(E)) =$	and	
tl:TemporalClass	C_I	ITC	ITC $\subseteq$ C_I	This defines ITC as the set of TL-OWL classes
tl:TemporalVariable		ITV	ITV $\subseteq$ C_I	This defines ITV as the set of TL-OWL temporal variables
tl:Qualification	C_I	ITQ	ITQ $\subseteq$ C_I	This defines ITQ as the set of TL-OWL qualifications

Table 7. Characteristics of TL-OWL classes and properties

If E is	then if e $\in$ $CEXT_I(S_I(E))$ then	Note
tl : TemporalClass	$CEXT_I(e) \subseteq IOT$	Instances of TL-OWL classes are TL-OWL individuals.
tl : TemporalVariable	$CEXT_I(e) \subseteq T$	TL-OWL temporal variables are time intervals.
If E is	then c $\in$ $CEXT_I(S_I(E))$ iff c $\in$ IOOP $\cup$ IODP and	Note
tl : Parametric-FunctionalProperty	$\langle x,y_1 \rangle$, $\langle x,y_2 \rangle \in EXT_I(c)$ implies $y_1 = y_2$, independently from time	Both individual-valued and datatype properties can be parametric functional properties.
If E is	then c $\in$ $CEXT_I(S_I(E))$ iff c $\in$ IOOP and	Note
tl : InverseParametric-FunctionalProperty	$\langle x_1,y \rangle$, $\langle x_2,y \rangle \in EXT_I(c)$ implies $x_1 = x_2$, independently from time	Individual-valued properties can be inverse parametric functional properties.

Table 8. Conditions on TL-OWL restrictions and qualifications

If	then x $\in$ IOR, y $\in$ IOC $\cup$ IDC, p $\in$ IOOP $\cup$ IODP, and $CEXT_I(x) =$
$\langle x,y \rangle \in EXT_I(S_I(tl:selectValuesFrom)) \wedge \langle x,p \rangle \in EXT_I(S_I(owl:onProperty))$	$\{u \in domp \mid EXT_I(p)(u) \in CEXT_I(y)\}$
If	then z $\in$ ITQ, d $\in$ IOC $\cup$ IDC, x,y $\in$ ITV, and $CEXT_I(z) =$
$\langle z,d \rangle \in EXT_I(S_I(tl:bindVariable)) \wedge \langle z,x \rangle \in EXT_I(S_I(tl:onVariable))$	$\{u \in IOT \mid u \in CEXT_I(d), t = V(x)\}$
$\langle z,d \rangle \in EXT_I(S_I(tl:bindSubstitutiveVariable)) \wedge \langle z,x \rangle \in EXT_I(S_I(tl:onVariable)) \wedge \langle z,y \rangle \in EXT_I(S_I(tl:onSubstitutiveVariable))$	$\{u \in IOT \mid u \in CEXT_I(d), H = H \cup \{y \rightarrow V(x)\}\}$

Table 9. Conditions on TL-OWL features

If E is	then $\langle x,y \rangle \in EXT_I(S_I(E))$ iff
tl : agreementOf	x$\in$IOC, y is a sequence of p,q over IOOP $\cup$ IODP, $CEXT_I(x) = \{u \in domp \cap domq \mid EXT_I(p)(u) = EXT_I(q)(u)\}$
tl : disagreementOf	x$\in$IOC, y is a sequence of p,q over IOOP $\cup$ IODP, $CEXT_I(x) = \{u \in domp \cap domq \mid EXT_I(p)(u) \neq EXT_I(q)(u)\}$
tl : undefinednessOf	x$\in$IOC, y is a partial function over IOOP $\cup$ IODP, $CEXT_I(x)$=IOT-domp
tl : pathOf	x$\in$IOOP, y is a sequence of p,q over IOOP, and u$\in$domp and v$\in$domq, $\langle u,v \rangle \in EXT_I(p) \cap \langle v,w \rangle \in EXT_I(q)$ implies $\langle u,w \rangle \in EXT_I(x)$

Table 10. Conditions on TL-OWL temporal relations

If E is	then $x \in CEXT_I(S_I(E))$ iff
tl : NOW	$CEXT_I(x)$ is the current interval
If E is	$\langle x,y \rangle \in EXT_I(S_I(E))$ iff
tl : before	$CEXT_I(x)=[u_1,v_1] \wedge CEXT_I(y)=[u_2,v_2]$ implies $v_1 < u_2$
other temporal relations ...	...

We now show that the two model theories, the direct model-theoretic semantics from Sect. 3.3 and the RDF-compatible model-theoretic semantics from this section, have the same consequences on TL-OWL ontologies that can be written in the abstract syntax.

Definition 8. *Let K and Q be collections of RDF graphs and D be a datatype map. Then K* **TL-OWL entails** *Q with respect to D iff every TL-OWL interpretation with respect to D (of any vocabulary V that includes the RDF and RDFS vocabularies and the TL-OWL vocabulary) that satisfies all the RDF graphs in K also satisfies all the RDF graphs in Q. K is* **TL-OWL consistent** *iff there is some TL-OWL interpretation that satisfies all the RDF graphs in K.*

Theorem 1. *Let O and O' be collections of TL-OWL ontologies and axioms and facts in abstract syntax form. Given a datatype map D that maps xsd:string and xsd:integer to the appropriate XML Schema datatypes and that includes the RDF mapping for rdf:XMLLiteral, then O entails O' with respect to D if and only if the translation of O TL-OWL entails the translation of O' with respect to D.*

Proof (sketch): This theorem can be proved by a structural induction for all of directives and constructs, but the description of its complete proof is too long. Therefore, in this paper we only introduce the outline of the proof due to the restriction of pages.

Given a datatype map D, a separated TL-OWL vocabulary is defined into a set of URI references V' = VO + VC + VTC + VTV + VD + VI + VOP + VDP + VAP + VXP. The translation of the separated TL-OWL vocabulary T(V') consists of all the triples of the form

v rdf:type owl:Ontology. v ∈ VO, v rdf:type owl:Class. v ∈ VC, v rdf:type tl:TemporalClass. v ∈ VTC, v rdf:type tl:TemporalVariable. v ∈ VTV, v rdf:type rdfs:Datatype. v ∈ VD, v rdf:type owl:Thing. v ∈ VI, v rdf:type owl:Object Property. v ∈ VOP, v rdf:type owl:DatatypeProperty. v ∈ VDP, v rdf:type owl:AnnotationProperty. v ∈ VAP, v rdf:type owl:OntologyProperty. v ∈ VXP.

Further, a collection of TL-OWL ontologies, axioms, and facts in abstract syntax form, O, with a separated vocabulary is defined with the new notion of a separated vocabulary V = VO + VC + VTC + VTV + VD + VI + VOP + VDP + VAP + VXP, where all URI references used as ontology names are taken from VO, class IDs are taken from VC, temporal class IDs are taken from VTC, temporal variable IDs are taken from VTV, datatype IDs are taken from VD, individual IDs are taken from VI, individual-valued property IDs are taken

from VOP, data-valued property IDs are taken from VDP, annotation property IDs are taken from VAP, and ontology property IDs are taken from VXP.

Then the above theorem can be paraphrased as the following : Let O and O' be collections of TL-OWL ontologies, axioms, and facts in abstract syntax form. Then O direct entails O' if and only if T(O) TL-OWL entails T(O').

In order to prove this theorem, first, we inductively check whether all the constructs of descriptions and qualifications and all the directives of TemporalClass, TemporalVariable, ObjectProperty and DatatypeProperty satisfy the above theorem.

Suppose O entails O'. Let I be a TL-OWL DL interpretation that satisfies T(O). Then from the above structural induction, there is some direct interpretation I' such that for any abstract TL-OWL ontology or axiom or fact X over V', I satisfies T(X) iff I' satisfies X. Thus I' satisfies each ontology in O. Because O entails O', I' satisfies O', so I satisfies T(O'). Thus T(K),T(V') TL-OWL DL entails T(Q). Conversely, suppose T(O) TL-OWL DL entails T(O'). Let I' be an direct interpretation that satisfies K. Then from the above structural induction, there is some TL-OWL DL interpretation I such that for any abstract TL-OWL ontology X over V', I satisfies T(X) iff I' satisfies X. Thus I satisfies T(O). Because T(O) TL-OWL DL entails T(O'), I satisfies T(O'), so I' satisfies O'. Thus O entails O'. Consequently, by the correspondence of two semantics, we can conclude that O direct entails O' if and only if T(O) TL-OWL entails T(O')

3.6 Reasoning in TL-OWL

Artale and Franconi present a subsumption reasoning for $\mathcal{TL}$-$\mathcal{ALCF}$. The calculus is based on the idea of separating the inference on the temporal part($\mathcal{TL}$) from the inference on the DL part($\mathcal{ALCF}$). This is achieved by first looking for a normal form of concepts. The normalization procedure generates a completed existential form of the form: $\Diamond(\overline{X})\overline{Tc}.(Q_0 \sqcap Q_1@X_1 \sqcap ... \sqcap Q_n@X_n)$, where each Q is a non-temporal concept, $\overline{X}$ is a set of temporal variable and $\overline{Tc}$ is a set of temporal constraints. Thanks to the completed existential form, concept subsumption in $\mathcal{TL}$-$\mathcal{ALCF}$ can be reduced to concept subsumption between non-temporal concepts and to subsumption between temporal constraint networks, i.e., a labeled directed graph $\langle \overline{X}, \overline{Tc}, \overline{Q@X} \rangle$, where arcs are labeled with a set of arbitrary temporal relationship and nodes are labeled with non-temporal concepts. Moreover, Artale and Franconi show that the subsumption problem in $\mathcal{TL}$-$\mathcal{ALCF}$ can be reduced to the subsumption between $\mathcal{ALCF}$ concepts, i.e., the non-temporal part (see Theorem 7.11 in [2]).

The temporal description logic to formalize TL-OWL is $\mathcal{TL}$-$\mathcal{SHOIN}(\mathbf{D})$, which extends the non-temporal part($\mathcal{ALCF}$) of $\mathcal{TL}$-$\mathcal{ALCF}$ to $\mathcal{SHOIN}(\mathbf{D})$. By this extension this logic can have the expressivity of OWL DL. Moreover, if the completed existential form for $\mathcal{TL}$-$\mathcal{SHOIN}(\mathbf{D})$ can be shown by the normalization procedure, the reduction of $\mathcal{TL}$-$\mathcal{ALCF}$ can be also used in $\mathcal{TL}$-$\mathcal{SHOIN}(\mathbf{D})$.

Most steps of the normalization procedure shown in $\mathcal{TL}$-$\mathcal{ALCF}$ is for the temporal part so that it can be sufficient that only simple form[1] of $\mathcal{TL}$-$\mathcal{SHOIN}(\mathbf{D})$

[1] Artale calls a negation normal form to a simple form.

is defined to obtain the completed existential form for $\mathcal{TL}$-$\mathcal{SHOIN}(\mathbf{D})$. We define the simple form excluding those of $\mathcal{TL}$-$\mathcal{ALCF}$ as follows. These are similar to simple forms presented in [7] and the simple form of $\mathcal{TL}$-$\mathcal{ALCF}$ is also presented in Fig. 14 of [2].

$\neg \leqslant nR.C \rightarrow \geqslant (n+1)R.C \qquad \neg\exists T.d \rightarrow \forall T.\neg d$
$\neg \geqslant (n+1)R.C \rightarrow \leqslant nR.C \qquad \neg\forall T.d \rightarrow \exists T.\neg d$
$\neg \geqslant 0R.C \rightarrow \ C \sqcap \neg C$

As for $\mathcal{TL}$-$\mathcal{ALCF}$ case, without loss of generality, the normalization procedure for $\mathcal{TL}$-$\mathcal{SHOIN}(\mathbf{D})$ reduces the *subsumption problem* in $\mathcal{TL}$-$\mathcal{SHOIN}(\mathbf{D})$ to the *subsumption* between $\mathcal{SHOIN}(\mathbf{D})$ concepts. It can be also shown that a $\mathcal{TL}$-$\mathcal{SHOIN}(\mathbf{D})$ concept in completed existential form, $\langle \overline{X}, \overline{Tc}, \overline{Q@X} \rangle$, is *satisfiable* if and only if the non-temporal concepts labeling each node in $\overline{X}$ are *satisfiable*. Moreover, following the above reduction, the subsumption problem of $\mathcal{TL}$-$\mathcal{SHOIN}(\mathbf{D})$ is decidable because that of $\mathcal{SHOIN}(\mathbf{D})$ is decidable. The proof is similar to the one for $\mathcal{TL}$-$\mathcal{ALCF}$ (see Proposition 7.8 and Theorem 7,11 in [2]) and we here do not mention it because it is straightforward.

4 Related Work

OWL-Time ontology [13] has recently been proposed to describe the temporal contents of Web pages as well as the temporal properties of Web services. The OWL-Time was formerly the DAML-Time and is currently published as the status of W3C working draft. The ontology provides a vocabulary to represent the topological relations among instants and intervals, along with information about duration and datetime. It is also shown how the ontology can be used within OWL-S by several examples. OWL-Time however is not an ontology language, but a time ontology based on OWL. It therefore have to use the rule-based reasoning to solve the problem as introduced in Sect. 1.

[11] and [8] present a new temporal ontology language based on OWL, which is a similar approach to ours. In order to represent time and temporal aspects such as change in ontologies, they introduce time slices (the temporal parts of an individual) and fluents (properties that hold between timeslices) and define their semantics based on Description Logic. They however do not show the formal reasoning algorithm such as subsumption and entailment. We can not also be convinced that the reasoning for the language is decidable.

5 Conclusions

OWL as an ontology language for Semantic Web can represent and reason the knowledge for information resources on web. It however cannot perform the reasoning for temporal information, since OWL cannot represent the temporal semantics. Therefore, we propose an interval-based temporal ontology language TL-OWL, which can represent and reason time information on Semantic Web. In

order to formalize TL-OWL the abstract and the exchange syntax of TL-OWL and their model semantics are defined. These two semantics are also shown to have the same consequences on TL-OWL ontologies that can be written in the abstract syntax. Finally, we show that the resoning in TL-OWL is decidable.

References

1. Allen, J.F.: Maintaining knowledge about temporal intervals. In: Communications of the ACM, vol. 26(11), pp. 832–843. ACM, New York (1983)
2. Artale, A., Franconi, E.: A Temporal Description Logic for Reasoning about Actions and Plans. Journal of Artificial Intelligence Research 9, 463–506 (1998)
3. Artale, A., Franconi, E.: A Survey of Temporal Extensions of Description Logics. Annals of Mathematics and Artificial Intelligence 4(1), 171–210 (2001)
4. Bettini, C.: Time dependent concepts: Representation and reasoning using temporal description logics. Data & Knowledge Engineering 22(1), 1–38 (1997)
5. Halpern, J.Y., Moses, Y.: A Propositional Modal Logic of Time Intervals. Journal of ACM 38(4), 935–962 (1991)
6. Hollunder, B., Nutt, W.: Subsumption Algorithms for Concept Languages, Technical Research Report RR-90-04, DFKI, Germany (1990)
7. Horrocks, I., Sattler, U.: A tableaux decision procedure for SHOIQ. In: Proc. of the 19th International Joint Conference on Artificial Intelligence, pp. 448–453 (2005)
8. Milea, V., Frasincar, F., Kaymak, U., Noia, T.: An OWL-based Approach Towards Representing Time in Web Information Systems. In: Proc. of the 4th International Workshop of Web Information Systems Modeling Workshop, pp. 791–802 (2007)
9. Schild, K.D.: Combining terminological logics with tense logic. In: Proc. of the 6th Portuguese Conference on Artificial Intelligence (1993)
10. Schmiedel, A.: A temporal terminological logic. In: Proc. of the AAAI 1990, pp. 640–645 (1990)
11. Welty, C., Fikes, R., Makarios, S.: A Reusable Ontology for Fluents in OWL. In: Proc. of the International Conference on Formal Ontology in Information Systems, pp. 226–236 (2006)
12. Wolter, F., Zakharyaschev, M.: Temporalizing description logics, Frontiers of Combining Systems. Studies Press-Wiley, Chichester (1999)
13. W3C Working Draft, Time Ontology in OWL (2006), `http://www.w3.org/TR/owl-time`