

# Temporal Classes and OWL

Natalya Keberle  
ZNU, Ukraine

# Outline

- Temporal object as a design primitive
- Examples
- Issues
- Solutions

# Temporal objects

- Model dynamic nature of the Universe of Discourse
- Evidently or latently use the time structure
- Modeled differently, no common recommendations
- No agreed language/reasoning support

# Examples

Assemble-Spaghetti-Marinara equivalent

exist  $y, z, w$  such that

*before* ( $y, w$ )

and *before* ( $z, w$ )

and (Boil-Spaghetti *at*  $y$ )

and (Make-Marinara *at*  $z$ )

and (Put-Together-SM *at*  $w$ )

$y, z, w$  – temporal intervals

Taken from (Artale&Franconi, JAIR, 1998)

# Examples

Student equivalent

(Person intersect

(*somepast* Entrant intersect

(*somefuture* Graduate

union

*somefuture* Failed)))

The life cycle of being a student...

# Temporal DL: why it's good

- Time-related issues in the Semantic Web applications are usually modeled with \*:
  - **Versioning** – each change of state of an object causes a new version
  - **4D-fluents** –property value supplied with temporal information objects are wrapped by time slices
  - **Temporal RDF-Graphs** reified with OWL-Time
- Such issues can be modeled with:
  - **Temporal DL** – keep all the temporal semantics inside the language constructs.
    - No additional actions while modeling– work with really temporal language

\* - arguments are discussed in (Baratis et al, SSTD, 2009)

# Issues with TDL: Time Structure

- Interval-based
- Point-based
- Various properties of time
  - Discrete/dense, linear/branching, finite/infinite, cyclic,...

Properties of time influences the complexity of reasoning in TDL

# Issues with TDL: Level of Interoperation

- Temporal operations are allowed only in front of concept definition

( *somepast* C ) subclassOf ( *somepast* D )

- Temporal operations are allowed in front of axioms

*allfuture* ( C subclassof D )

The allowed scope of temporal operations application influences the complexity of reasoning in TDL

# Issues with TDL: Temporal & Non-temporal things in one ontology

- Classification
- How `tdl:TemporalThing` will be related to `owl:Thing`

# OWL-MeT

- Point-based, linear, infinite, discrete time line
- Temporal operations applicable to concepts only (no temporalized roles, no temporalized axioms)
- Underlying logic is MT-ALCO
- MT – Metric Time

# OWL-MeT

- For  $E, F$  – non-temporal,  $C, D$  – temporal concepts

$E, F \rightarrow A \mid top \mid bottom \mid E \sqcap F \mid E \sqcup F \mid \neg E \mid \exists R. E \mid \forall R. E \mid \{o\}$

$C, D \rightarrow E \mid \{a\} \mid C \text{ intersection } D \mid C \text{ union } D \mid \text{not } C \mid C@ \{a\} \mid \text{future } n C \mid$   
 $\mid \text{past } n C \mid \text{somefuture } C \mid \text{somepast } C \mid \text{allfuture } C \mid \text{allpast } C$

- Temporal formulae are

$C \text{ equivalent } D, C \text{ subclassof } D$

- Temporal formulae also are

$\varphi \text{ union } \psi, \varphi \text{ intersection } \psi, \text{not } \varphi$

# OWL-MeT

- Abstract and exchange syntax are available at <http://ermolayev.com/owl-met/>
- Reasoning support is provided with Pellet-MeT <http://ermolayev.com/owl-met/reasoner.htm>

## Outline of extensions

- Tableau algorithm was extended for temporal operations
- owl:Class subclassof owlmet:TClass
- new rdf:properties for owlmet:past, owlmet:allpast, owlmet:at,...

# OWL-MeT examples

A student is:

```
<TClass rdf:ID="Entrant"/>
<TClass rdf:ID="Graduated"/>
<TClass rdf:ID="Student">
  <equivalentClass>
    <intersectionOf>
      <TRestriction>
        <somepast rdf:resource="#Entrant"/>
      </TRestriction>
      <TRestriction>
        <allfuture>
          <TClass>
            <unionOf>
              <TClass about="#Student"/>
              <TClass about="#Graduated"/>
            </unionOf>
          </TClass>
        </allfuture>
      </TRestriction>
    </intersectionOf>
  </equivalentClass>
</TClass>
```

A first-year student is:

```
<TClass ID="Entrant"/>
<TClass rdf:ID="Student">
  <rdfs:subClassOf>
    <TRestriction>
      <past rdf:datatype=
        "&xsd;#NonNegativeInteger">
        1 </past>
      <equivalentClass>
        <TClass rdf:about="#Entrant"/>
      </equivalentClass>
    </TRestriction>
  </rdfs:subClassOf>
</TClass>
```