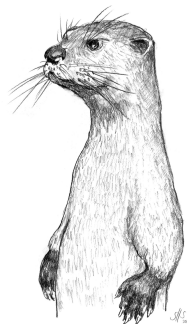


Reasonable Ontology Templates

Martin G. Skjæveland Henrik Forssell
Johan W. Klüwer Daniel Lupp
Evgenij Thorstensen Arild Waaler

WOP2017
October 21, 2017

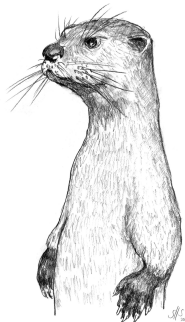


OTTR

Reasonable Ontology Templates (OTTR)

Introduction and plan for talk

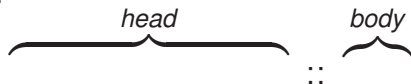
- Tools and methodology
- A better way of constructing ontologies
- OWL macros expressed in OWL
 - abstractions, encapsulation, clear interface
 - nested
- template: $\langle parameters \rangle \mapsto \mathcal{O}$
- instance: $\langle arguments \rangle \mapsto \mathcal{O}_{arguments}$
- OWL serialisation
 - OWL reasoning
 - Leverage W3C stack and tools
- Extensible framework for representing and transforming data
- Demo



OTTR

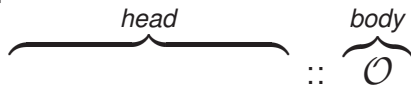
Ontology templates

- Template:



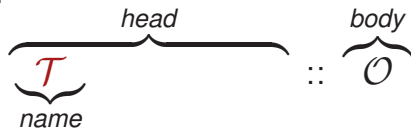
Ontology templates

- Template:



Ontology templates

- Template:



Ontology templates

- Template:

$$\underbrace{\mathcal{T}}_{\text{name}} \underbrace{(\underbrace{p_1, \dots, p_n}_{\text{parameters}})}_{\text{head}} :: \underbrace{\mathcal{O}}_{\text{body}}$$

Ontology templates

- Template:

$$\underbrace{\mathcal{T}}_{\text{name}} \underbrace{(p_1, \dots, p_n)}_{\text{parameters}} \text{ } \overbrace{\hspace{1cm}}^{\text{head}} :: \overbrace{\{p_1 \sqsubseteq \mathcal{C}, \hspace{1cm}\}}^{\text{body}}$$

- Parameters can be any concept, role, or individual name or data value

Ontology templates

- Template:

$$\underbrace{\mathcal{T}}_{\text{name}} \underbrace{(p_1, \dots, p_n)}_{\text{parameters}} \overset{\text{head}}{\text{::}} \overset{\text{body}}{\{p_1 \sqsubseteq C, D \sqsubseteq \exists p_2.E, \}}$$

- Parameters can be any concept, role, or individual name or data value

Ontology templates

- Template:

$$\underbrace{\mathcal{T}}_{\text{name}} \underbrace{(p_1, \dots, p_n)}_{\text{parameters}} \overset{\text{head}}{\text{::}} \overset{\text{body}}{\{p_1 \sqsubseteq C, D \sqsubseteq \exists p_2.E, F(p_3)\}}$$

- Parameters can be any concept, role, or individual name or data value

Ontology templates

- Template:

$$\underbrace{\mathcal{T}}_{\text{name}} \underbrace{(p_1, \dots, p_n)}_{\text{parameters}} \overset{\text{head}}{\text{::}} \overset{\text{body}}{\{p_1 \sqsubseteq C, D \sqsubseteq \exists p_2.E, F(p_3)\}}$$

- Parameters can be any concept, role, or individual name or data value
- Template instance:

$$\mathcal{T}_1 \underbrace{(p_1, p_2, p_3)}_{\text{arguments}} \Rightarrow \{p_1 \sqsubseteq C, D \sqsubseteq \exists p_2.E, F(p_3)\}$$

Ontology templates

- Template:

$$\underbrace{\mathcal{T}}_{\text{name}} \underbrace{(p_1, \dots, p_n)}_{\text{parameters}} \overset{\text{head}}{\text{::}} \overset{\text{body}}{\{p_1 \sqsubseteq C, D \sqsubseteq \exists p_2.E, F(p_3), \mathcal{T}_2(p_4, \dots)\}}$$

- Parameters can be any concept, role, or individual name or data value
- Template instance:

$$\mathcal{T}_1 \underbrace{(p_1, p_2, p_3)}_{\text{arguments}} \Rightarrow \{p_1 \sqsubseteq C, D \sqsubseteq \exists p_2.E, F(p_3)\}$$

- Templates can be (non-cyclically) nested

Ontology templates

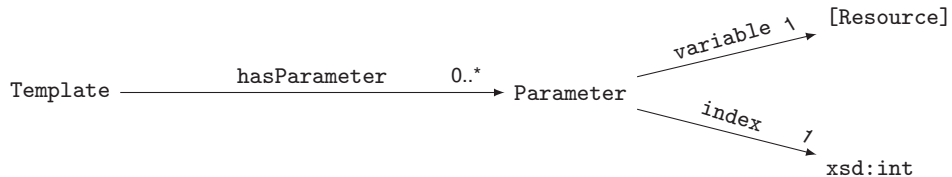
- Template:

$$\underbrace{\mathcal{T}}_{\text{name}} \underbrace{(p_1, \dots, p_n)}_{\text{parameters}} \overset{\text{head}}{\text{::}} \overset{\text{body}}{\{p_1 \sqsubseteq C, D \sqsubseteq \exists p_2.E, F(p_3), \mathcal{T}_2(p_4, \dots)\}}$$

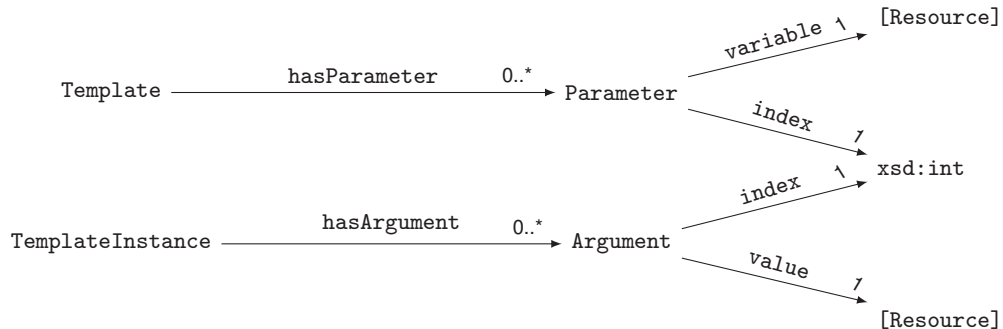
- Parameters can be any concept, role, or individual name or data value
- Template instance:

$$\mathcal{T}_1 \underbrace{(p_1, p_2, p_3)}_{\text{arguments}} \Rightarrow \{p_1 \sqsubseteq C, D \sqsubseteq \exists p_2.E, F(p_3), \mathcal{T}_2(\dots)\}$$

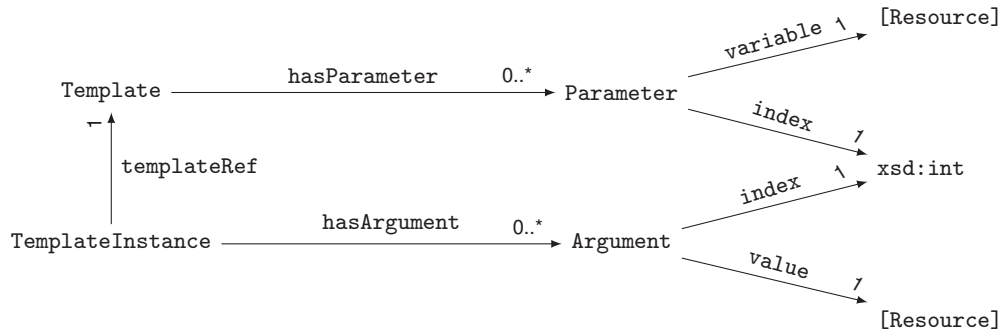
- Templates can be (non-cyclically) nested
- Instances are expanded recursively



<http://ns.ottr.xyz>

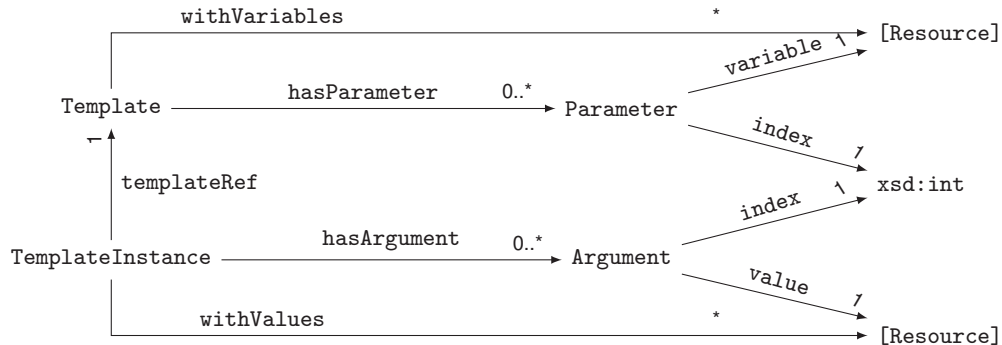


<http://ns.ottr.xyz>



<http://ns.ottr.xyz>

OTTR OWL vocabulary



<http://ns.ottr.xyz>

PartOf(*Whole*, *Part*) :: { *Whole* $\sqsubseteq$ $\exists$ *hasPart*.*Part* }

PartOf(*Whole*, *Part*) :: { *Whole* $\sqsubseteq$ $\exists$ *hasPart*.*Part* }

PartOf(*Hammer*, *Handle*)

PartOf(*Whole*, *Part*) :: { *Whole* $\sqsubseteq$ $\exists$ *hasPart.Part* }

PartOf(*Hammer*, *Handle*) $\Rightarrow$
{ *Hammer* $\sqsubseteq$ $\exists$ *hasPart.Handle* }

PartOf(*Whole*, *Part*) :: { *Whole* $\sqsubseteq \exists$ *hasPart*.*Part* }

head:

```
<http://draft.ottr.xyz/i17/partof> a ottr:Template ;  
    ottr:hasParameter [ ottr:index 1; ottr:variable :Whole ] ,  
                      [ ottr:index 2; ottr:variable :Part ] .
```

body:

```
:Part a owl:Class .  
:Whole a owl:Class ;  
    rdfs:subClassOf [ a owl:Restriction ;  
        owl:onProperty ex:hasPart ; owl:someValuesFrom :Part ] .
```

PartOf(*Whole*, *Part*) :: { *Whole* $\sqsubseteq \exists$ *hasPart.Part* }

head:

```
<http://draft.ottr.xyz/i17/partof> a ottr:Template ;  
    ottr:hasParameter [ ottr:index 1; ottr:variable :Whole ] ,  
                      [ ottr:index 2; ottr:variable :Part ] .
```

body:

```
:Part a owl:Class .  
:Whole a owl:Class ;  
    rdfs:subClassOf [ a owl:Restriction ;  
        owl:onProperty ex:hasPart ; owl:someValuesFrom :Part ] .
```

PartOf(*Hammer*, *Handle*)

PartOf(*Whole*, *Part*) :: { *Whole* $\sqsubseteq \exists$ *hasPart.Part* }

head:

```
<http://draft.ottr.xyz/i17/partof> a ottr:Template ;  
    ottr:hasParameter [ ottr:index 1; ottr:variable :Whole ] ,  
                      [ ottr:index 2; ottr:variable :Part ] .
```

body:

```
:Part a owl:Class .  
:Whole a owl:Class ;  
    rdfs:subClassOf [ a owl:Restriction ;  
        owl:onProperty ex:hasPart ; owl:someValuesFrom :Part ] .
```

PartOf(*Hammer*, *Handle*)

```
[] ottr:templateRef <http://draft.ottr.xyz/i17/partof> ;  
    ottr:hasArgument [ ottr:index 1; ottr:value ex:Hammer ] ,  
                    [ ottr:index 2; ottr:value ex:Handle ] .
```

PartOf(*Whole*, *Part*) :: { *Whole* $\sqsubseteq$ $\exists$ *hasPart.Part* }

```
### head:
<http://draft.ottr.xyz/i17/partof> a ottr:Template ;
    ottr:hasParameter [ ottr:index 1; ottr:variable :Whole ] ,
                      [ ottr:index 2; ottr:variable :Part ] .
## ottr:withVariables ( :Whole :Part ) .

### body:
:Part a owl:Class .
:Whole a owl:Class ;
    rdfs:subClassOf [ a owl:Restriction ;
        owl:onProperty ex:hasPart ; owl:someValuesFrom :Part ] .
```

PartOf(*Hammer*, *Handle*)

```
[] ottr:templateRef <http://draft.ottr.xyz/i17/partof> ;
    ottr:hasArgument [ ottr:index 1; ottr:value ex:Hammer ] ,
                    [ ottr:index 2; ottr:value ex:Handle ] .
## ottr:withValues ( ex:Hammer ex:Handle ) .
```

PartOf(*Whole*, *Part*) :: { *SubSomeValuesFrom*(*Whole*, *hasPart*, *Part*) }

```
### head:
<http://draft.ottr.xyz/i17/partof> a ottr:Template ;
    ottr:hasParameter [ ottr:index 1; ottr:variable :Whole ] ,
                      [ ottr:index 2; ottr:variable :Part ] .
    ## ottr:withVariables ( :Whole :Part ) .

### body:
[] ottr:templateRef ottr-owl:SubSomeValuesFrom ;
    ottr:withValues ( :Whole ex:hasPart :Part ) .
```

PartOf(*Hammer*, *Handle*)

```
[] ottr:templateRef <http://draft.ottr.xyz/i17/partof> ;
    ottr:hasArgument [ ottr:index 1; ottr:value ex:Hammer ] ,
                     [ ottr:index 2; ottr:value ex:Handle ] .
    ## ottr:withValues ( ex:Hammer ex:Handle ) .
```

PartOf(*Whole*, *Part*) :: { *SubSomeValuesFrom*(*Whole*, *hasPart*, *Part*) }

- *RDF* macros
- not only *OWL*
- use for *LOD*

```
### head:
<http://draft.ottr.xyz/i17/partof> a ottr:Template ;
    ottr:hasParameter [ ottr:index 1; ottr:variable :Whole ] ,
                      [ ottr:index 2; ottr:variable :Part ] .
    ## ottr:withVariables ( :Whole :Part ) .

### body:
[] ottr:templateRef ottr-owl:SubSomeValuesFrom ;
    ottr:withValues ( :Whole ex:hasPart :Part ) .
```

PartOf(*Hammer*, *Handle*)

```
[] ottr:templateRef <http://draft.ottr.xyz/i17/partof> ;
    ottr:hasArgument [ ottr:index 1; ottr:value ex:Hammer ] ,
                     [ ottr:index 2; ottr:value ex:Handle ] .
    ## ottr:withValues ( ex:Hammer ex:Handle ) .
```

PartOf(*Whole*, *Part*) :: { *SubSomeValuesFrom*(*Whole*, *hasPart*, *Part*) }

```
### head:
<http://draft.ottr.xyz/i17/partof> a ottr:Template ;
    ottr:hasParameter [ ottr:index 1; ottr:variable :Whole ] ,
                      [ ottr:index 2; ottr:variable :Part ] .
    ## ottr:withVariables ( :Whole :Part ) .

### body:
[] ottr:templateRef ottr-owl:SubSomeValuesFrom ;
    ottr:withValues ( :Whole ex:hasPart :Part ) .
```

PartOf(*Hammer*, *Handle*)

```
[] ottr:templateRef <http://draft.ottr.xyz/i17/partof> ;
    ottr:hasArgument [ ottr:index 1; ottr:value ex:Hammer ] ,
                     [ ottr:index 2; ottr:value ex:Handle ] .
    ## ottr:withValues ( ex:Hammer ex:Handle ) .
```

- *RDF* macros
 - not only *OWL*
 - use for *LOD*
- Expansion:
 - copy template graph
 - paragraph $\mapsto$ argument
 - recurse

PartOf(*Whole*, *Part*) :: { *SubSomeValuesFrom*(*Whole*, *hasPart*, *Part*) }

```
### head:
<http://draft.ottr.xyz/i17/partof> a ottr:Template ;
    ottr:hasParameter [ ottr:index 1; ottr:variable :Whole ] ,
                      [ ottr:index 2; ottr:variable :Part ] .
    ## ottr:withVariables ( :Whole :Part ) .

### body:
[] ottr:templateRef ottr-owl:SubSomeValuesFrom ;
    ottr:withValues ( :Whole ex:hasPart :Part ) .
```

PartOf(*Hammer*, *Handle*)

```
[] ottr:templateRef <http://draft.ottr.xyz/i17/partof> ;
    ottr:hasArgument [ ottr:index 1; ottr:value ex:Hammer ] ,
                     [ ottr:index 2; ottr:value ex:Handle ] .
    ## ottr:withValues ( ex:Hammer ex:Handle ) .
```

- *RDF* macros
 - not only *OWL*
 - use for *LOD*
- Expansion:
 - copy template graph
 - paragraph $\mapsto$ argument
 - recurse
- Arguments
 - any *RDF* resource
 - incl. e.g., *RDF(S)/OWL* voc.
 - template IRIs
 - lists
 - variable-length args
 - multiple instances

PartOf(*Whole*, *Part*) :: { *SubSomeValuesFrom*(*Whole*, *hasPart*, *Part*) }

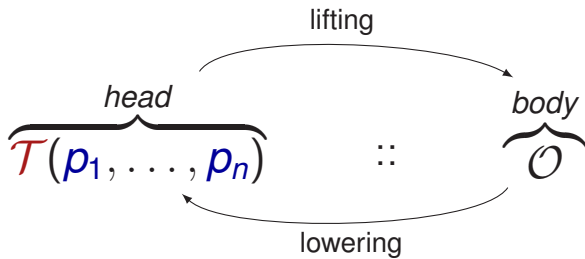
```
### head:
<http://draft.ottr.xyz/i17/partof> a ottr:Template ;
    ottr:hasParameter [ ottr:index 1; ottr:variable :Whole ] ,
                      [ ottr:index 2; ottr:variable :Part ] .
## ottr:withVariables ( :Whole :Part ) .

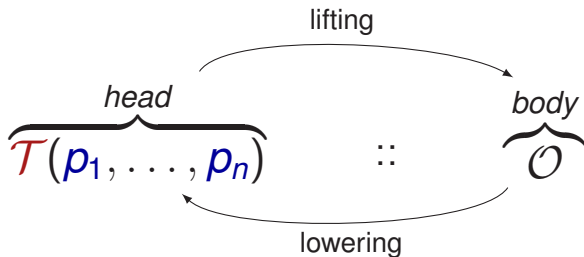
### body:
[] ottr:templateRef ottr-owl:SubSomeValuesFrom ;
    ottr:withValues ( :Whole ex:hasPart :Part ) .
```

PartOf(*Hammer*, *Handle*)

```
[] ottr:templateRef <http://draft.ottr.xyz/i17/partof> ;
    ottr:hasArgument [ ottr:index 1; ottr:value ex:Hammer ] ,
                     [ ottr:index 2; ottr:value ex:Handle ] .
## ottr:withValues ( ex:Hammer ex:Handle ) .
```

- *RDF* macros
 - not only *OWL*
 - use for *LOD*
- Expansion:
 - copy template graph
 - paragraph $\mapsto$ argument
 - recurse
- Arguments
 - any *RDF* resource
 - incl. e.g., *RDF(S)/OWL* voc.
 - template IRIs
 - lists
 - variable-length args
 - multiple instances
- Reasoning
 - body
 - head
 - expanded
 - instantiated





head instance	head format	lifting & lowering	body format
XML, XSLx	XSD+SAWSDL	XSLT	OWL
OTTRs (RDF)	(SHACL, ShEx)	SPARQL	OWL

All formats generated from templates!

- <http://www.ottr.xyz>

Reasonable Ontology Templates (OTTR)

Reasonable Ontology Templates, OTTRs or Templates for short, are OWL ontology macros capable of representing ontology design patterns and closely integrating their use into ontology engineering. A template is itself an RDF graph or an OWL ontology, annotated with a special purpose OWL vocabulary. This allows templates to be edited, debugged, published, identified, instantiated, composed, used as queries and bulk transformations, and maintained; all leveraging existing W3C standards, best practices and tools.

An OTTR consists of a head and a body. The head specifies a tabular input format in the form of a list of parameters, and the body contains a set of RDF triples or OWL axioms, which may use the parameters of the head as ordinary RDF resources. A template is instantiated by substituting the parameters with arguments. Templates can be nested; a template may contain an instance to a different template in its body and pass parameter values to it. Such complex templates can be expanded by recursively replacing the template instance with the body it represents.

[OTTR vocabulary](#)

OTTRs are represented using RDF graphs or OWL ontologies, by marking the RDF graph as containing a template and marking RDF resources as parameters. This is done using a special purpose OWL vocabulary, the [OTTR ontology](#).

[Multiple formats](#)

A template can be seen as a mapping between a tabular format (the template head) to the a graph or ontology structure (the body). We can exploit this by representing these formats and transformations between them, in both directions, using existing W3C standards.

The list of currently available formats are found in the top right menu on the information pages for each individual template, see e.g., [partlength](#).

[Library of Reasonable Ontology Templates](#)

The list of available OTTRs hosted at this site can be browsed in the [templates library](#).

Templates are organised into different repositories according to status and use.

You can share templates in the [draft](#) repository, an open git repository (you will need to create a (free) account at GELab to be able to push to the repository).

[Implementation](#)

A prototype implementation that can interpret the [templates vocabulary](#), perform the necessary substitution and expansion, and serve templates on various formats, is available as a set of Java servlets hosted at <http://owl.ottr.xyz/>. (This specific link provides no additional information, but in fact redirects to this page). This implementation generates the information page for a template, e.g., [partlength](#) and the other formats found on this page, by reading the templates published at the above repositories.

In fact, the implementation can read any template available online. The link to the template must be specified in the URI with the URI parameter `tpl`, e.g., <http://owl.ottr.xyz/info?tpl=http://draft.ottr.xyz/LI/partlength>. Feel free to use the service to test your own templates.

In other to test templates locally, a feature limited executable Java jar, [oslottr.jar](#) is available for download and for testing purposes only. It supports expanding templates and any RDF graph containing template calls. Files can be accessed online or locally.

Quick links
• [home](#)
• [vocabulary](#)
• [library](#)
• [g5](#)



- <http://www.ottr.xyz>
 - CLI java tool

```
File Edit View Search Terminal Help
pizzatoppings.rdf:type owl:Class .

## Generated by the Owl API (version 5.1.0) https://github.com/owls/owlapi/
##temp/ottr.js java -jar owlapi.jar -expand -in pizzat.ttl

[Prefix : <http://example.com/> .
Prefix p: <http://example.com/external/> .
Prefix owl: <http://www.w3.org/2002/07/owl#> .
Prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
Prefix owl: <http://www.w3.org/2000/01/rdf-schema#> .
Prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
Base <http://www.w3.org/2002/07/owl#> .

( rdf:type owl:Ontology
) .

#####
# Object Properties
#####

## http://example.com/external/hasTopping
p:hasTopping rdf:type owl:ObjectProperty .

#####
# Classes
#####

## http://example.com/AnchoviesTopping
AnchoviesTopping rdf:type owl:Class .

## http://example.com/CaperTopping
CaperTopping rdf:type owl:Class .

## http://example.com/FourSeasons
FourSeasons rdf:type owl:Class .
rdfs:subClassOf p:NamedPizza
[ rdf:type owl:Restriction ;
  owl:property p:hasTopping ;
  owl:someValuesFrom AnchoviesTopping
]
[ rdf:type owl:Restriction ;
  owl:property p:hasTopping ;
  owl:someValuesFrom CaperTopping
]
[ rdf:type owl:Restriction ;
  owl:property p:hasTopping ;
  owl:someValuesFrom MozzarellaTopping
]
[ rdf:type owl:Restriction ;
  owl:property p:hasTopping ;
  owl:someValuesFrom MushroomTopping
]
```

- <http://www.ottr.xyz>
 - CLI java tool
- OTTR vocabulary:
<http://ns.ottr.xyz>

Reasonable Ontology Templates (OTTRs) Vocabulary

[templates-lite](#)

The templates-lite ontology introduces the minimal vocabulary for using and defining templates. However, it is primarily useful for specifying template instances as it contains no axioms to check the consistency of template specification.

Latest version

<http://ns.ottr.xyz/templates-lite.owl>

Documentation

<http://ns.ottr.xyz/templates-lite.html>

Versions

- 0.2: <http://ns.ottr.xyz/version/templates-lite-0.2.owl>
- 0.1: <http://ns.ottr.xyz/version/template-instances-0.1.owl>

[templates-core](#)

The templates-core vocabulary extends the templates-lite ontology by adding more properties and axioms for specifying and validating templates.

Latest version

<http://ns.ottr.xyz/templates-core.owl>

Documentation

<http://ns.ottr.xyz/templates-core.html>

Versions

- 0.2: <http://ns.ottr.xyz/version/templates-core-0.1.owl>
- 0.1: <http://ns.ottr.xyz/version/templates-0.1.owl>

[Shape Expression for validating legal templates RDF graphs](#)

Latest version

[templates.shexc](#)



- <http://www.ottr.xyz>
 - CLI java tool
- OTTR vocabulary:
<http://ns.ottr.xyz>
- Library of OTTRs:
<http://library.ottr.xyz>

Repositories • Candidate ◦ OWL ◦ ODPs • Draft • Test candidate/OWL • atom ◦ Cardinality ◦ DataCardinality ◦ ListRelation ◦ ObjectCardinality ◦ TypedListRelation ◦ ValueRestriction • axiom ◦ DifferentIndividuals ◦ DisjointClasses ◦ DisjointProperties ◦ DisjointUnion ◦ EquivalentValuesFrom ◦ EquivalentAllValuesFrom ◦ EquivalentExactCardinality ◦ EquivalentDataHasValue ◦ EquivalentDataMaxCardinality ◦ EquivalentDataMinCardinality ◦ EquivalentDataSomeValuesFrom ◦ EquivalentExactCardinality ◦ EquivalentHasValue ◦ EquivalentMaxCardinality ◦ EquivalentMinCardinality ◦ EquivalentObjectAllValuesFrom ◦ EquivalentObjectExactCardinality ◦ EquivalentObjectHasValue ◦ EquivalentObjectIntersectionOf ◦ EquivalentObjectMaxCardinality ◦ EquivalentObjectMinCardinality ◦ EquivalentObjectOneOf ◦ EquivalentObjectSomeValuesFrom ◦ EquivalentObjectUnionOf ◦ EquivalentSomeValuesFrom ◦ EquivalentClass ◦ EquivalentDataProperty	Library of Reasonable Ontology Templates (OTTRs) Reasonable Ontology Templates See www.ottr.xyz for more information. Repositories This OTTR library uses different repositories for storing templates according to their quality and status. standard: gilt not yet in use candidate: contents - gilt • OWL contains OTTRs that most of the OWL expressions listed in Mapping from the Structural Specification to RDF Graphs. • ODP, an acronym for Ontology Design Patterns, see http://ontologydesignpatterns.org, contains OTTR representations of ODPs. draft: contents - gilt Playground for testing and developing templates. tests: contents - gilt Contains templates that as used for testing template implementations. These are designed to give various parsing and expansion errors. The OTTR template library is available in a frames version. 
---	---

- <http://www.ottr.xyz>
 - CLI java tool
- OTTR vocabulary:
<http://ns.ottr.xyz>
- Library of OTTRs:
<http://library.ottr.xyz>
- Online web application:
<http://osl.ottr.xyz>

Repositories

- Candidate
 - OWL
 - ODPs
- Draft
- Test

draft

- chess
 - AgentRole.2
 - ChessGame.ttl
 - Event.2
 - List.4
 - BlackPlayerAgent
 - ChessGame
 - ChessGameReport
 - ChessGameReportExample
 - Player
 - Tournament
 - WhitePlayerAgent
- i17
 - partlength
 - partof
 - qualityvalue
- i18
 - partof
- o2
 - equipment
 - example
 - exampleInstance1
 - exampleInstance2
 - partof
 - qualityRange
- p12
 - AnObjectsMassIn kilogram.ttl
 - AnObjectsQuantityDatum.ttl
 - HammerExample.ttl
 - QuantityDatum.ttl
- pizza
 - NamedPizza

ObjectUnionOf

IRI: <http://candidate.ottr.xyz/owl/restrictionObjectUnionOf>

Arguments:

1. nonLiteralVariable: 2c23759bf69790b63bd25e24d92ef2ec
2. listVariable: 2db96aac15e949a635a5937

SubObjectSomeValuesFrom

IRI: <http://candidate.ottr.xyz/owl/axiom/SubObjectSomeValuesFrom>

Arguments:

1. classVariable: pizza
2. objectPropertyVariable: p:hasTopping
3. classVariable: 2db96aac15e949a635a5937

Diagram

RDF graph visualisation of the expanded body:

Source: <http://draft.ottr.xyz/pizza/NamedPizza>

Compiled:

```

@prefix p: <http://example.com/external/> .
@prefix r: <http://draft.ottr.xyz/pizza/NamedPizza/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix ottr: <http://ns.ottr.xyz/templated/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

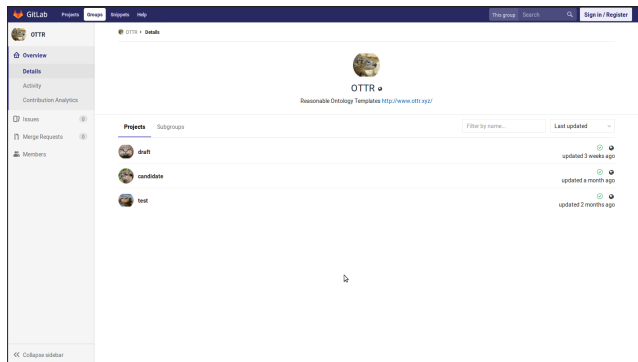
```

Available formats

- Specification
 - all
 - head
 - body
 - expansion
- Expansion
 - all
 - body
 - head
- Queries
 - select head
 - select body
 - insert_construct
 - insert_update
 - insert_delete
 - insert_delete_update
- Format Sample
 - XML format
 - XML schema

Resources

- <http://www.ottr.xyz>
 - CLI java tool
- OTTR vocabulary:
<http://ns.ottr.xyz>
- Library of OTTRs:
<http://library.ottr.xyz>
- Online web application:
<http://osl.ottr.xyz>
- Git:
<http://gitlab.com/ottr/>



- <http://www.ottr.xyz>
 - CLI java tool
- OTTR vocabulary:
<http://ns.ottr.xyz>
- Library of OTTRs:
<http://library.ottr.xyz>
- Online web application:
<http://osl.ottr.xyz>
- Git:
<http://gitlab.com/ottr/>
- ISWC Poster

Reasonable Ontology Templates

Marin G. Skjæveland¹ Henrik Forssell¹ Johan W. Kåre² Daniel P. Lupo³ Evgenij Thorstensen¹ Arild Waaler¹
¹Department of Informatics, University of Oslo, ²DNV GL, Norway

Overview

- Modular macros for OWL/RDF — in OWL/RDF
 - Good for ontology construction and maintenance
 - Clear interface, consistent pattern use, hide complexity
 - Reasoning support
 - DRY — don't repeat yourself
- Leverage W3C stack and tools
- Implementation, OWL vocabulary, and template library available at <http://ottr.xyz>

Preliminaries

An (ontology) template has a head and a body:

$$\begin{array}{c} \text{head} \\ \text{Template} \left(\begin{array}{c} \text{P}_1, \dots, \text{P}_n \end{array} \right) \quad \text{body} \\ \text{parameters} \end{array} \quad \text{O}$$

- A template instance $T(\text{P}_1, \dots, \text{P}_n)$ specifies a substitution of the template's parameters in the body with the instance's arguments.
- The template body is a regular ontology that can contain template instances and where parameters may be used as placeholders for concepts, roles, individuals and data values.
- A template instance is expanded by recursively replacing template instances with substituted template bodies.

Extensible Framework

$$T(\text{P}_1, \dots, \text{P}_n) \xrightleftharpoons[\text{lowering}]{\text{lifting}} \text{O}$$

- Bulk data formats: XLSx, XML (XSD + SAWSDL), RDF, OWL
- Bulk lifting and lowering transformations: XSLT, SPARQL

OTTR OWL Vocabulary: <http://ns.ottr.xyz>

Examples

- Template `PartOf(Whole, Part) :: { Whole ⊆ ∃ hasPart . Part }` serialised with OTTR vocabulary:

```
## head:
<draft:ottr:xyz/117/partof> a ottr:Template ;
ottr:hasParameter [ ottr:index 1; ottr:value :Whole ] ,
[ ottr:index 2; ottr:value :Part ] .
## or: ottr:withVariables ( :Whole :Part ) .
## body:
:Part a owl:Class .
:Whole a owl:Class ;
rdfs:subClassOf [ a owl:Restriction ;
owl:hasProperty ex:hasPart ; owl:someValuesFrom :Part ] .
```
- Template instance `PartOf(Hammer, Handle)` serialised with OTTR vocabulary:

```
[ ottr:templateRef <draft:ottr:xyz/117/partof> ;
ottr:hasArgument [ ottr:index 1; ottr:value :Hammer ] ,
[ ottr:index 2; ottr:value :Handle ] ] .
## or: ottr:withValues ( ex:Hammer ex:Handle ) .
```
- ...expands to

```
:Handle a owl:Class .
:Hammer a owl:Class ;
rdfs:subClassOf [ a owl:Restriction ;
owl:hasProperty ex:hasPart ; owl:someValuesFrom :Handle ] .
```

Examples

- Tense input and iterations with lists + use of 3 "official" OTTRs:

```
<draft:ottr:xyz/pizza/NamedPizza> a ottr:Template ;
ottr:withVariables ( :pizza ( :toppings ) ) .
## body:
:pizza rdfs:subClassOf p:NamedPizza .
[ ottr:templateRef ottr-owl:SubObjectSomeValuesFrom ;
ottr:hasArgument [ ottr:index 1; ottr:value :pizza ] ,
[ ottr:index 2; ottr:value p:hasTopping ] ,
[ ottr:index 3; ottr:eachValue ( :toppings ) ] ] .
[ ottr:templateRef ottr-owl:SubObjectAllValuesFrom ;
ottr:withValues ( :pizza p:hasTopping _alltoppings ) ] .
[ ottr:templateRef ottr-owl:ObjectIntersection ;
ottr:withValues ( _alltoppings ( :toppings ) ) ] .
```
- The instance `NamedPizza(Napoletana, (Tomato, Mozzarella, Olive, Capar, Anchovies))` ... expands to

```
Napoletana ⊑ NamedPizza
Napoletana ⊑ V hasTopping ( Tomato ⊔ Mozzarella ⊔ ... )
Napoletana ⊑ ∃ hasTopping.Tomato
Napoletana ⊑ ∃ hasTopping.Mozzarella
Napoletana ⊑ ∃ hasTopping...
```

Future Work

- Large-scale evaluation
 - prototype successfully tested in industry
- Language design and extensions
- Support template pre-conditions
 - with also template head as ontology
- Methods for template library maintenance
 - with ontology interrelationships
- Protégé plugin
- Template-based visualisations

UiO : University of Oslo

The Future and Future work

- Engineering ontologies using only OTTR templates
- API for OWL (— or AI for OWL)
- Ontology experts construct OTTRs
- “Knowledge engineers” combine OTTRs to user-facing templates
- Domain experts use tabular format

The Future and Future work

- Engineering ontologies using only OTTR templates
- API for OWL (— or AI for OWL)
- Ontology experts construct OTTRs
- “Knowledge engineers” combine OTTRs to user-facing templates
- Domain experts use tabular format
- Large-scale evaluation
- Further development of templates and tools
- Theory for structuring a library templates, e.g., what is a subtemplate?
- Template library maintenance
- Ontology as pre-condition: template heads as ontologies too
- Protege plugin
- Excel plugin