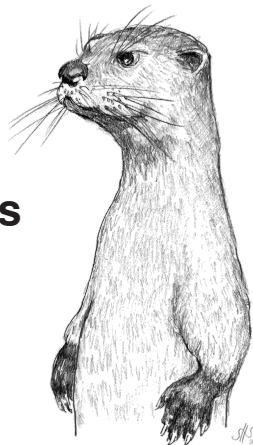


Reasonable Ontology Templates (OTTR)

Martin G. Skjæveland, Daniel P. Lupp,
Leif Harald Karlsen and Henrik Forssell



Improving the efficiency and quality of ontology engineering

Problem:

- Current ontology engineering methods are too low-level
- Makes ontology construction repetitive and error-prone
- Difficult to engage end-users
- Difficult to generate sustainable large ontologies
- Difficult to efficiently maintain ontologies

Improving the efficiency and quality of ontology engineering

Problem:

- Current ontology engineering methods are too low-level
- Makes ontology construction repetitive and error-prone
- Difficult to engage end-users
- Difficult to generate sustainable large ontologies
- Difficult to efficiently maintain ontologies

We need:

- Better abstractions!
- Capture and instantiate reoccurring modelling patterns
- Formats adapted to domain experts, data managers and ontology experts

OTTR templates example 1

```
p:Margherita rdf:type owl:Class .  
p:Margherita rdfs:subClassOf p:Pizza .  
p:Margherita rdfs:label "Margherita"@it .
```

```
p:Hawaii rdf:type owl:Class .  
p:Hawaii rdfs:subClassOf p:Pizza .  
p:Hawaii rdfs:label "Hawaii"@en .
```

```
p:Grandiosa rdf:type owl:Class .  
p:Grandiosa rdfs:subClassOf p:Pizza .  
p:Grandiosa rdfs:label "Grandiosa"@no .
```

OTTR templates example 1

```
p:Margherita rdf:type owl:Class .  
p:Margherita rdfs:subClassOf p:Pizza .  
p:Margherita rdfs:label "Margherita"@it .  
  
p:Hawaii rdf:type owl:Class .  
p:Hawaii rdfs:subClassOf p:Pizza .  
p:Hawaii rdfs:label "Hawaii"@en .  
  
p:Grandiosa rdf:type owl:Class .  
p:Grandiosa rdfs:subClassOf p:Pizza .  
p:Grandiosa rdfs:label "Grandiosa"@no .
```

OTTR templates example 1

```
p:Margherita rdf:type owl:Class .  
p:Margherita rdfs:subClassOf p:Pizza .  
p:Margherita rdfs:label "Margherita"@it .
```

```
p:Hawaii rdf:type owl:Class .  
p:Hawaii rdfs:subClassOf p:Pizza .  
p:Hawaii rdfs:label "Hawaii"@en .
```

```
p:Grandiosa rdf:type owl:Class .  
p:Grandiosa rdfs:subClassOf p:Pizza .  
p:Grandiosa rdfs:label "Grandiosa"@no .
```

```
?name rdf:type owl:Class .  
      rdfs:subClassOf p:Pizza .  
?name rdfs:label ?label .
```

OTTR templates example 1

```
p:Margherita rdf:type owl:Class .  
p:Margherita rdfs:subClassOf p:Pizza .  
p:Margherita rdfs:label "Margherita"@it .
```

```
p:Hawaii rdf:type owl:Class .  
p:Hawaii rdfs:subClassOf p:Pizza .  
p:Hawaii rdfs:label "Hawaii"@en .
```

```
p:Grandiosa rdf:type owl:Class .  
p:Grandiosa rdfs:subClassOf p:Pizza .  
p:Grandiosa rdfs:label "Grandiosa"@no .
```

```
PIZZA(?name, ?label) ::  
    ?name rdf:type owl:Class .  
    rdfs:subClassOf p:Pizza .  
    ?name rdfs:label ?label .
```

OTTR templates example 1

```
p:Margherita rdf:type owl:Class .  
p:Margherita rdfs:subClassOf p:Pizza .  
p:Margherita rdfs:label "Margherita"@it .
```

```
p:Hawaii rdf:type owl:Class .  
p:Hawaii rdfs:subClassOf p:Pizza .  
p:Hawaii rdfs:label "Hawaii"@en .
```

```
p:Grandiosa rdf:type owl:Class .  
p:Grandiosa rdfs:subClassOf p:Pizza .  
p:Grandiosa rdfs:label "Grandiosa"@no .
```

```
PIZZA(?name, ?label) ::  
  TRIPLE(?name, rdf:type, owl:Class),  
  TRIPLE(, rdfs:subClassOf, p:Pizza),  
  TRIPLE(?name, rdfs:label, ?label) .
```


OTTR templates example 1

```
p:Margherita rdf:type owl:Class .  
p:Margherita rdfs:subClassOf p:Pizza .  
p:Margherita rdfs:label "Margherita"@it .
```

```
p:Hawaii rdf:type owl:Class .  
p:Hawaii rdfs:subClassOf p:Pizza .  
p:Hawaii rdfs:label "Hawaii"@en .
```

```
p:Grandiosa rdf:type owl:Class .  
p:Grandiosa rdfs:subClassOf p:Pizza .  
p:Grandiosa rdfs:label "Grandiosa"@no .
```

```
PIZZA(?name, ?label) ::  
  TRIPLE(?name, rdf:type, owl:Class),  
  TRIPLE(, rdfs:subClassOf, p:Pizza),  
  TRIPLE(?name, rdfs:label, ?label) .
```

```
PIZZA(p:Margherita, "Margherita"@it) .
```

```
PIZZA(p:Hawaii, "Hawaii"@en) .
```

```
PIZZA(p:Grandiosa, "Grandiosa"@no) .
```

OTTR templates example 1

```
p:Margherita rdf:type owl:Class .
p:Margherita rdfs:subClassOf p:Pizza .
p:Margherita rdfs:label "Margherita"@it .

p:Hawaii rdf:type owl:Class .
p:Hawaii rdfs:subClassOf p:Pizza .
p:Hawaii rdfs:label "Hawaii"@en .

p:Grandiosa rdf:type owl:Class .
p:Grandiosa rdfs:subClassOf p:Pizza .
p:Grandiosa rdfs:label "Grandiosa"@no .
```

```
SUBCLASSOF(?sub, ?super) ::
    TRIPLE(?sub, rdfs:subClassOf, ?super) .

PIZZA(?name, ?label) ::
    TRIPLE(?name, rdf:type, owl:Class),
    TRIPLE(, rdfs:subClassOf, p:Pizza),
    TRIPLE(?name, rdfs:label, ?label) .

PIZZA(p:Margherita, "Margherita"@it) .
PIZZA(p:Hawaii, "Hawaii"@en) .
PIZZA(p:Grandiosa, "Grandiosa"@no) .
```

OTTR templates example 1

```
p:Margherita rdf:type owl:Class .
p:Margherita rdfs:subClassOf p:Pizza .
p:Margherita rdfs:label "Margherita"@it .

p:Hawaii rdf:type owl:Class .
p:Hawaii rdfs:subClassOf p:Pizza .
p:Hawaii rdfs:label "Hawaii"@en .

p:Grandiosa rdf:type owl:Class .
p:Grandiosa rdfs:subClassOf p:Pizza .
p:Grandiosa rdfs:label "Grandiosa"@no .
```

```
SUBCLASSOF(?sub, ?super) ::
    TRIPLE(?sub, rdfs:subClassOf, ?super) .

PIZZA(?name, ?label) ::
    TRIPLE(?name, rdf:type, owl:Class),
    SUBCLASSOF(?name, p:Pizza),
    TRIPLE(?name, rdfs:label, ?label) .

PIZZA(p:Margherita, "Margherita"@it) .
PIZZA(p:Hawaii, "Hawaii"@en) .
PIZZA(p:Grandiosa, "Grandiosa"@no) .
```

OTTR templates example 1

```
p:Margherita rdf:type owl:Class .  
p:Margherita rdfs:subClassOf p:Pizza .  
p:Margherita rdfs:label "Margherita"@it .
```

- Abstractions

```
SUBCLASSOF(?sub, ?super) ::  
  TRIPLE(?sub, rdfs:subClassOf, ?super) .
```

```
PIZZA(?name, ?label) ::  
  TRIPLE(?name, rdf:type, owl:Class),  
  SUBCLASSOF(?name, p:Pizza),  
  TRIPLE(?name, rdfs:label, ?label) .
```

```
PIZZA(p:Margherita, "Margherita"@it) .  
PIZZA(p:Hawaii, "Hawaii"@en) .  
PIZZA(p:Grandiosa, "Grandiosa"@no) .
```

OTTR templates example 1

```
p:Margherita rdf:type owl:Class .  
p:Margherita rdfs:subClassOf p:Pizza .  
p:Margherita rdfs:label "Margherita"@it .
```

- Abstractions
- Don't Repeat Yourself (DRY)

```
SUBCLASSOF(?sub, ?super) ::  
  TRIPLE(?sub, rdfs:subClassOf, ?super) .
```

```
PIZZA(?name, ?label) ::  
  TRIPLE(?name, rdf:type, owl:Class),  
  SUBCLASSOF(?name, p:Pizza),  
  TRIPLE(?name, rdfs:label, ?label) .
```

```
PIZZA(p:Margherita, "Margherita"@it) .  
PIZZA(p:Hawaii, "Hawaii"@en) .  
PIZZA(p:Grandiosa, "Grandiosa"@no) .
```

OTTR templates example 1

```
p:Margherita rdf:type owl:Class .  
p:Margherita rdfs:subClassOf p:Pizza .  
p:Margherita rdfs:label "Margherita"@it .
```

- Abstractions
- Don't Repeat Yourself (DRY)
- Modular patterns
- Encapsulate complexity

```
SUBCLASSOF(?sub, ?super) ::  
  TRIPLE(?sub, rdfs:subClassOf, ?super) .
```

```
PIZZA(?name, ?label) ::  
  TRIPLE(?name, rdf:type, owl:Class),  
  SUBCLASSOF(?name, p:Pizza),  
  TRIPLE(?name, rdfs:label, ?label) .
```

```
PIZZA(p:Margherita, "Margherita"@it) .  
PIZZA(p:Hawaii, "Hawaii"@en) .  
PIZZA(p:Grandiosa, "Grandiosa"@no) .
```

OTTR templates example 1

```
p:Margherita rdf:type owl:Class .  
p:Margherita rdfs:subClassOf p:Pizza .  
p:Margherita rdfs:label "Margherita"@it .
```

- Abstractions
- Don't Repeat Yourself (DRY)
- Modular patterns
- Encapsulate complexity
- Uniform modelling

```
SUBCLASSOF(?sub, ?super) ::  
    TRIPLE(?sub, rdfs:subClassOf, ?super) .
```

```
PIZZA(?name, ?label) ::  
    TRIPLE(?name, rdf:type, owl:Class),  
    SUBCLASSOF(?name, p:Pizza),  
    TRIPLE(?name, rdfs:label, ?label) .
```

```
PIZZA(p:Margherita, "Margherita"@it) .  
PIZZA(p:Hawaii, "Hawaii"@en) .  
PIZZA(p:Grandiosa, "Grandiosa"@no) .
```

OTTR templates example 1

```
p:Margherita rdf:type owl:Class .  
p:Margherita rdfs:subClassOf p:Pizza .  
p:Margherita rdfs:label "Margherita"@it .
```

- Abstractions
- Don't Repeat Yourself (DRY)
- Modular patterns
- Encapsulate complexity
- Uniform modelling
- Separate design and content

```
SUBCLASSOF(?sub, ?super) ::  
    TRIPLE(?sub, rdfs:subClassOf, ?super) .
```

```
PIZZA(?name, ?label) ::  
    TRIPLE(?name, rdf:type, owl:Class),  
    SUBCLASSOF(?name, p:Pizza),  
    TRIPLE(?name, rdfs:label, ?label) .
```

```
PIZZA(p:Margherita, "Margherita"@it) .  
PIZZA(p:Hawaii, "Hawaii"@en) .  
PIZZA(p:Grandiosa, "Grandiosa"@no) .
```


OTTR templates example 1

```
p:Margherita rdf:type owl:Class .  
p:Margherita rdfs:subClassOf p:Pizza .  
p:Margherita rdfs:label "Margherita"@it .
```

- Abstractions
- Don't Repeat Yourself (DRY)
- Modular patterns
- Encapsulate complexity
- Uniform modelling
- Separate design and content
- Ensure completeness of input
- Simple input format

```
SUBCLASSOF(?sub, ?super) ::  
    TRIPLE(?sub, rdfs:subClassOf, ?super) .
```

```
PIZZA(?name, ?label) ::  
    TRIPLE(?name, rdf:type, owl:Class),  
    SUBCLASSOF(?name, p:Pizza),  
    TRIPLE(?name, rdfs:label, ?label) .
```

```
PIZZA(p:Margherita, "Margherita"@it) .  
PIZZA(p:Hawaii, "Hawaii"@en) .  
PIZZA(p:Grandiosa, "Grandiosa"@no) .
```

OTTR templates example 1

```
p:Margherita rdf:type owl:Class .  
p:Margherita rdfs:subClassOf p:Pizza .  
p:Margherita rdfs:label "Margherita"@it .
```

- Abstractions
- Don't Repeat Yourself (DRY)
- Modular patterns
- Encapsulate complexity
- Uniform modelling
- Separate design and content
- Ensure completeness of input
- Simple input format
- Substitution
- Macro expansion

```
SUBCLASSOF(?sub, ?super) ::  
    TRIPLE(?sub, rdfs:subClassOf, ?super) .
```

```
PIZZA(?name, ?label) ::  
    TRIPLE(?name, rdf:type, owl:Class),  
    SUBCLASSOF(?name, p:Pizza),  
    TRIPLE(?name, rdfs:label, ?label) .
```

```
PIZZA(p:Margherita, "Margherita"@it) .  
PIZZA(p:Hawaii, "Hawaii"@en) .  
PIZZA(p:Grandiosa, "Grandiosa"@no) .
```

OTTR templates example 1

```
p:Margherita rdf:type owl:Class .  
p:Margherita rdfs:subClassOf p:Pizza .  
p:Margherita rdfs:label "Margherita"@it .
```

- Abstractions
- Don't Repeat Yourself (DRY)
- Modular patterns
- Encapsulate complexity
- Uniform modelling
- Separate design and content
- Ensure completeness of input
- Simple input format
- Substitution
- Macro expansion

```
SUBCLASSOF(?sub, ?super) ::  
    TRIPLE(?sub, rdfs:subClassOf, ?super) .
```

```
PIZZA(?name, ?label) ::  
    TRIPLE(?name, rdf:type, owl:Class),  
    SUBCLASSOF(?name, p:Pizza),  
    TRIPLE(?name, rdfs:label, ?label) .
```

```
PIZZA(p:Margherita, "Margherita"@it) .  
PIZZA(p:Hawaii, "Hawaii"@en) .  
PIZZA(p:Grandiosa, "Grandiosa"@no) .
```

OTTR templates example 1

```
p:Margherita rdf:type owl:Class .  
p:Margherita rdfs:subClassOf p:Pizza .  
p:Margherita rdfs:label "Margherita"@it .
```

- Abstractions
- Don't Repeat Yourself (DRY)
- Modular patterns
- Encapsulate complexity
- Uniform modelling
- Separate design and content
- Ensure completeness of input
- Simple input format
- Substitution
- Macro expansion

```
SUBCLASSOF(?sub, ?super) ::  
  TRIPLE(?sub, rdfs:subClassOf, ?super) .
```

```
PIZZA(?name, ?label) ::  
  TRIPLE(p:Margherita, rdf:type, owl:Class),  
  SUBCLASSOF(p:Margherita, p:Pizza),  
  TRIPLE(p:Margherita, rdfs:label, "Margherita"@it) .
```

```
PIZZA(p:Margherita, "Margherita"@it) .  
PIZZA(p:Hawaii, "Hawaii"@en) .  
PIZZA(p:Grandiosa, "Grandiosa"@no) .
```

OTTR templates example 1

```
p:Margherita rdf:type owl:Class .  
p:Margherita rdfs:subClassOf p:Pizza .  
p:Margherita rdfs:label "Margherita"@it .
```

- Abstractions
- Don't Repeat Yourself (DRY)
- Modular patterns
- Encapsulate complexity
- Uniform modelling
- Separate design and content
- Ensure completeness of input
- Simple input format
- Substitution
- Macro expansion

```
SUBCLASSOF(?sub, ?super) ::  
  TRIPLE(?sub, rdfs:subClassOf, ?super) .
```

```
PIZZA(?name, ?label) ::  
  TRIPLE(p:Margherita, rdf:type, owl:Class),  
  SUBCLASSOF(p:Margherita, p:Pizza),  
  TRIPLE(p:Margherita, rdfs:label, "Margherita"@it) .
```

```
PIZZA(p:Margherita, "Margherita"@it) .  
PIZZA(p:Hawaii, "Hawaii"@en) .  
PIZZA(p:Grandiosa, "Grandiosa"@no) .
```

OTTR templates example 1

```
p:Margherita rdf:type owl:Class .  
p:Margherita rdfs:subClassOf p:Pizza .  
p:Margherita rdfs:label "Margherita"@it .
```

- Abstractions
- Don't Repeat Yourself (DRY)
- Modular patterns
- Encapsulate complexity
- Uniform modelling
- Separate design and content
- Ensure completeness of input
- Simple input format
- Substitution
- Macro expansion

```
SUBCLASSOF(?sub, ?super) ::  
    TRIPLE(p:Margherita, rdfs:subClassOf, p:Pizza) .  
  
PIZZA(?name, ?label) ::  
    TRIPLE(p:Margherita, rdf:type, owl:Class),  
    SUBCLASSOF(p:Margherita, p:Pizza),  
    TRIPLE(p:Margherita, rdfs:label, "Margherita"@it) .  
  
PIZZA(p:Margherita, "Margherita"@it) .  
PIZZA(p:Hawaii, "Hawaii"@en) .  
PIZZA(p:Grandiosa, "Grandiosa"@no) .
```

OTTR templates example 1

```
p:Margherita rdf:type owl:Class .  
p:Margherita rdfs:subClassOf p:Pizza .  
p:Margherita rdfs:label "Margherita"@it .
```

- Abstractions
- Don't Repeat Yourself (DRY)
- Modular patterns
- Encapsulate complexity
- Uniform modelling
- Separate design and content
- Ensure completeness of input
- Simple input format
- Substitution
- Macro expansion

```
SUBCLASSOF(?sub, ?super) ::  
    TRIPLE(p:Margherita, rdfs:subClassOf, p:Pizza) .  
  
PIZZA(?name, ?label) ::  
    TRIPLE(p:Margherita, rdf:type, owl:Class),  
    TRIPLE(p:Margherita, rdfs:subClassOf, p:Pizza),  
    TRIPLE(p:Margherita, rdfs:label, "Margherita"@it) .  
  
PIZZA(p:Margherita, "Margherita"@it) .  
PIZZA(p:Hawaii, "Hawaii"@en) .  
PIZZA(p:Grandiosa, "Grandiosa"@no) .
```

Pizza Ontology

- Pizza ontology tutorial
- 22 NamedPizza-s
- Difficult to find all instances—understanding the ontology
- Error-prone to update pattern

The screenshot shows the Protege OWL editor interface. The top bar displays the URL: <http://www.c-o-d-e.org/ontologies/pizza/pizza.2.0.0>. The main window is divided into several panes:

- Left Pane (Class Hierarchy):** Shows a tree view of the ontology. The 'pizza' class is expanded, showing its subclasses: 'DomainThing', 'Country', 'Food', 'IceCream', 'Pizza', 'CheesyPizza', 'InterestingPizza', 'MeatyPizza', 'NamedPizza', 'American', 'AmericanHot', 'Cajun', 'Capricciosa', 'Caprina', 'Fiorentina', 'FourSeasons', 'FruttiDiMare', 'Giardiniera', 'LaReine', 'Margherita' (highlighted in blue), 'Mushroom', 'Napoletana', 'Parmense', 'PolloAdAstra', 'PrinceCarlo', 'QuattroFormaggi', 'Rosa', 'Siciliana', 'SloppyGiuseppe', 'Soho', 'Veneziana', 'NonVegetarianPizza', 'RealItalianPizza', 'SpicyPizza', 'SpicyPizzaEquivalent', 'ThinAndCrispyPizza', and 'UnclosedPizza'.
- Top Pane (Annotations):** Shows the 'Margherita' class selected. The 'Annotations' tab is active, displaying a list of annotations for 'Margherita' in English and Portuguese, including 'rdfs:label', 'skos:prefLabel', and 'skos:altLabel'.
- Bottom Pane (Description):** Shows the 'Description' tab for 'Margherita'. It displays the class's description, including 'hasTopping only (MozzarellaTopping or TomatoTopping)', 'hasTopping some MozzarellaTopping', 'hasTopping some TomatoTopping', and 'NamedPizza'.

The status bar at the bottom indicates 'No Reasoner set. Select a reasoner from the Reasoner menu.' and 'Show Inference' is checked.

OTTR templates example 2

```
Margherita  $\sqsubseteq$  NamedPizza  
Margherita  $\sqsubseteq \exists$  hasCountryOfOrigin.{Italy}  
Margherita  $\sqsubseteq \forall$  hasTopping.(Tomato  $\sqcup$  Mozzarella)  
Margherita  $\sqsubseteq \exists$  hasTopping.Tomato  
Margherita  $\sqsubseteq \exists$  hasTopping.Mozzarella
```

OTTR templates example 2

```
Margherita  $\sqsubseteq$  NamedPizza  
Margherita  $\sqsubseteq \exists$  hasCountryOfOrigin.{Italy}  
Margherita  $\sqsubseteq \forall$  hasTopping.(Tomato  $\sqcup$  Mozzarella)  
Margherita  $\sqsubseteq \exists$  hasTopping.Tomato  
Margherita  $\sqsubseteq \exists$  hasTopping.Mozzarella
```

```
NAMEDPIZZA( ?Name : 1 class, ?Country : ? individual, ?Toppings : + class) |
```

OTTR templates example 2

```
Margherita  $\sqsubseteq$  NamedPizza  
Margherita  $\sqsubseteq$   $\exists$  hasCountryOfOrigin.{Italy}  
Margherita  $\sqsubseteq$   $\forall$  hasTopping.(Tomato  $\sqcup$  Mozzarella)  
Margherita  $\sqsubseteq$   $\exists$  hasTopping.Tomato  
Margherita  $\sqsubseteq$   $\exists$  hasTopping.Mozzarella
```

template name

```
NAMEDPIZZA( ?Name : 1 class, ?Country : ? individual, ?Toppings : + class) |
```

OTTR templates example 2

```
Margherita  $\sqsubseteq$  NamedPizza  
Margherita  $\sqsubseteq \exists$  hasCountryOfOrigin.{Italy}  
Margherita  $\sqsubseteq \forall$  hasTopping.(Tomato  $\sqcup$  Mozzarella)  
Margherita  $\sqsubseteq \exists$  hasTopping.Tomato  
Margherita  $\sqsubseteq \exists$  hasTopping.Mozzarella
```

template name
NAMEDPIZZA(?Name : 1 class, ?Country : ? individual, ?Toppings : + class)

OTTR templates example 2

```
Margherita  $\sqsubseteq$  NamedPizza  
Margherita  $\sqsubseteq$   $\exists$  hasCountryOfOrigin.{Italy}  
Margherita  $\sqsubseteq$   $\forall$  hasTopping.(Tomato  $\sqcup$  Mozzarella)  
Margherita  $\sqsubseteq$   $\exists$  hasTopping.Tomato  
Margherita  $\sqsubseteq$   $\exists$  hasTopping.Mozzarella
```

template name parameter name
 mandatory
 type
NAMEDPIZZA(?Name : 1 class, ?Country : ? individual, ?Toppings : + class) |

OTTR templates example 2

```
Margherita  $\sqsubseteq$  NamedPizza  
Margherita  $\sqsubseteq$   $\exists$  hasCountryOfOrigin.{Italy}  
Margherita  $\sqsubseteq$   $\forall$  hasTopping.(Tomato  $\sqcup$  Mozzarella)  
Margherita  $\sqsubseteq$   $\exists$  hasTopping.Tomato  
Margherita  $\sqsubseteq$   $\exists$  hasTopping.Mozzarella
```

template name parameter name mandatory type optional input

NAMEDPIZZA(?Name : 1 class, ?Country : ? individual, ?Toppings : + class) |

OTTR templates example 2

```
Margherita  $\sqsubseteq$  NamedPizza  
Margherita  $\sqsubseteq$   $\exists$  hasCountryOfOrigin.{Italy}  
Margherita  $\sqsubseteq$   $\forall$  hasTopping.(Tomato  $\sqcup$  Mozzarella)  
Margherita  $\sqsubseteq$   $\exists$  hasTopping.Tomato  
Margherita  $\sqsubseteq$   $\exists$  hasTopping.Mozzarella
```

template name parameter name mandatory type optional input list input

NAMEDPIZZA(?Name : 1 class, ?Country : ? individual, ?Toppings : + class)

OTTR templates example 2

```
Margherita  $\sqsubseteq$  NamedPizza  
Margherita  $\sqsubseteq \exists$  hasCountryOfOrigin.{Italy}  
Margherita  $\sqsubseteq \forall$  hasTopping.(Tomato  $\sqcup$  Mozzarella)  
Margherita  $\sqsubseteq \exists$  hasTopping.Tomato  
Margherita  $\sqsubseteq \exists$  hasTopping.Mozzarella
```

template name parameter name mandatory type optional input list input

```
NAMEDPIZZA( ?Name : 1 class, ?Country : ? individual, ?Toppings : + class )  
:: SUBCLASSOF(?Name, :NamedPizza),  
  SUBOBJECTHASVALUE(?Name, :hasCountryOfOrigin, ?Country),  
  SUBOBJECTALLVALUESFROM(?Name, :hasTopping, _:b1),  
    OBJECTUNIONOF(_:b1, ?Toppings),  
  x | SUBOBJECTSOMEVALUESFROM(?Name, :hasTopping, ?Toppings) .
```


OTTR templates example 2

```
Margherita  $\sqsubseteq$  NamedPizza
Margherita  $\sqsubseteq$   $\exists$  hasCountryOfOrigin.{Italy}
Margherita  $\sqsubseteq$   $\forall$  hasTopping.(Tomato  $\sqcup$  Mozzarella)
Margherita  $\sqsubseteq$   $\exists$  hasTopping.Tomato
Margherita  $\sqsubseteq$   $\exists$  hasTopping.Mozzarella
```

template name parameter name mandatory type optional input list input

```
NAMEDPIZZA( ?Name : 1 class, ?Country : ? individual, ?Toppings : + class)
::  SUBCLASSOF(?Name, :NamedPizza),
    SUBOBJECTHASVALUE(?Name, :hasCountryOfOrigin, ?Country),
    SUBOBJECTALLVALUESFROM(?Name, :hasTopping, _:b1),
    OBJECTUNIONOF(_:b1, ?Toppings),
    x | SUBOBJECTSOMEVALUESFROM(?Name, :hasTopping, ?Toppings) .
```

$?Name \sqsubseteq$:NamedPizza

OTTR templates example 2

```
Margherita  $\sqsubseteq$  NamedPizza
Margherita  $\sqsubseteq$   $\exists$  hasCountryOfOrigin.{Italy}
Margherita  $\sqsubseteq$   $\forall$  hasTopping.(Tomato  $\sqcup$  Mozzarella)
Margherita  $\sqsubseteq$   $\exists$  hasTopping.Tomato
Margherita  $\sqsubseteq$   $\exists$  hasTopping.Mozzarella
```

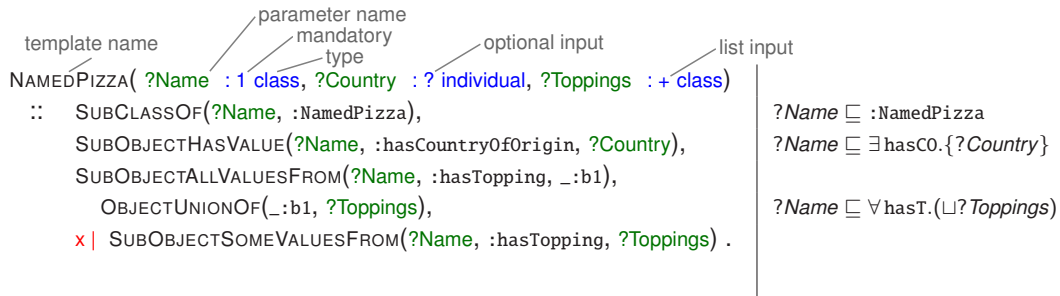
template name parameter name mandatory type optional input list input

```
NAMEDPIZZA( ?Name : 1 class, ?Country : ? individual, ?Toppings : + class )
::  SUBCLASSOF(?Name, :NamedPizza),
    SUBOBJECTHASVALUE(?Name, :hasCountryOfOrigin, ?Country),
    SUBOBJECTALLVALUESFROM(?Name, :hasTopping, _:b1),
    OBJECTUNIONOF(_:b1, ?Toppings),
    x | SUBOBJECTSOMEVALUESFROM(?Name, :hasTopping, ?Toppings) .
```

$?Name \sqsubseteq$:NamedPizza
 $?Name \sqsubseteq \exists$ hasC0.{?Country}

OTTR templates example 2

```
Margherita  $\sqsubseteq$  NamedPizza
Margherita  $\sqsubseteq$   $\exists$  hasCountryOfOrigin.{Italy}
Margherita  $\sqsubseteq$   $\forall$  hasTopping.(Tomato  $\sqcup$  Mozzarella)
Margherita  $\sqsubseteq$   $\exists$  hasTopping.Tomato
Margherita  $\sqsubseteq$   $\exists$  hasTopping.Mozzarella
```



OTTR templates example 2

```
Margherita  $\sqsubseteq$  NamedPizza
Margherita  $\sqsubseteq$   $\exists$  hasCountryOfOrigin.{Italy}
Margherita  $\sqsubseteq$   $\forall$  hasTopping.(Tomato  $\sqcup$  Mozzarella)
Margherita  $\sqsubseteq$   $\exists$  hasTopping.Tomato
Margherita  $\sqsubseteq$   $\exists$  hasTopping.Mozzarella
```

template name parameter name mandatory type optional input list input

```
NAMEDPIZZA( ?Name : 1 class, ?Country : ? individual, ?Toppings : + class )
::  SUBCLASSOF(?Name, :NamedPizza),
    SUBOBJECTHASVALUE(?Name, :hasCountryOfOrigin, ?Country),
    SUBOBJECTALLVALUESFROM(?Name, :hasTopping, _:b1),
    OBJECTUNIONOF(_:b1, ?Toppings),
    x | SUBOBJECTSOMEVALUESFROM(?Name, :hasTopping, ?Toppings) .
```

```
?Name  $\sqsubseteq$  :NamedPizza
?Name  $\sqsubseteq$   $\exists$  hasC0.{?Country}

?Name  $\sqsubseteq$   $\forall$  hasT.( $\sqcup$ ?Toppings)
?Name  $\sqsubseteq$   $\exists$  hasT.?Toppings...
```

OTTR templates example 2

```
Margherita  $\sqsubseteq$  NamedPizza
Margherita  $\sqsubseteq$   $\exists$  hasCountryOfOrigin.{Italy}
Margherita  $\sqsubseteq$   $\forall$  hasTopping.(Tomato  $\sqcup$  Mozzarella)
Margherita  $\sqsubseteq$   $\exists$  hasTopping.Tomato
Margherita  $\sqsubseteq$   $\exists$  hasTopping.Mozzarella
```

template name parameter name mandatory type optional input list input

```
NAMEDPIZZA( ?Name : 1 class, ?Country : ? individual, ?Toppings : + class )
::  SUBCLASSOF(?Name, :NamedPizza),
    SUBOBJECTHASVALUE(?Name, :hasCountryOfOrigin, ?Country),
    SUBOBJECTALLVALUESFROM(?Name, :hasTopping, _:b1),
    OBJECTUNIONOF(_:b1, ?Toppings),
    x | SUBOBJECTSOMEVALUESFROM(?Name, :hasTopping, ?Toppings) .
```

list expansion

```
?Name  $\sqsubseteq$  :NamedPizza
?Name  $\sqsubseteq$   $\exists$  hasC0.{?Country}

?Name  $\sqsubseteq$   $\forall$  hasT.( $\sqcup$ ?Toppings)
?Name  $\sqsubseteq$   $\exists$  hasT.?Toppings ...
```

OTTR templates example 2

```
Margherita  $\sqsubseteq$  NamedPizza  
Margherita  $\sqsubseteq \exists$  hasCountryOfOrigin.{Italy}  
Margherita  $\sqsubseteq \forall$  hasTopping.(Tomato  $\sqcup$  Mozzarella)  
Margherita  $\sqsubseteq \exists$  hasTopping.Tomato  
Margherita  $\sqsubseteq \exists$  hasTopping.Mozzarella
```

NAMEDPIZZA(Margherita, Italy, (Tomato, Mozzarella))

template name parameter name mandatory optional input list input

type

```
NAMEDPIZZA( ?Name : 1 class, ?Country : ? individual, ?Toppings : + class )  
:: SUBCLASSOF(?Name, :NamedPizza),  
   SUBOBJECTHASVALUE(?Name, :hasCountryOfOrigin, ?Country),  
   SUBOBJECTALLVALUESFROM(?Name, :hasTopping, _:b1),  
   OBJECTUNIONOF(_:b1, ?Toppings),  
   x | SUBOBJECTSOMEVALUESFROM(?Name, :hasTopping, ?Toppings) .
```

list expansion

```
?Name  $\sqsubseteq$  :NamedPizza  
?Name  $\sqsubseteq \exists$  hasC0.{?Country}  
  
?Name  $\sqsubseteq \forall$  hasT.( $\sqcup$ ?Toppings)  
?Name  $\sqsubseteq \exists$  hasT.?Toppings ...
```

OTTR templates example 2

```
Margherita  $\sqsubseteq$  NamedPizza
Margherita  $\sqsubseteq$   $\exists$  hasCountryOfOrigin. {Italy}
Margherita  $\sqsubseteq$   $\forall$  hasTopping. (Tomato  $\sqcup$  Mozzarella)
Margherita  $\sqsubseteq$   $\exists$  hasTopping. Tomato
Margherita  $\sqsubseteq$   $\exists$  hasTopping. Mozzarella
```

```
NAMEDPIZZA(Margherita, Italy, (Tomato, Mozzarella))
NAMEDPIZZA(Grandiosa, ottr:none, (Tomato, Jarlsberg, Ham, SweetPepper))
```

template name parameter name mandatory type optional input list input

```
NAMEDPIZZA( ?Name : 1 class, ?Country : ? individual, ?Toppings : + class )
::  SUBCLASSOF(?Name, :NamedPizza),
    SUBOBJECTHASVALUE(?Name, :hasCountryOfOrigin, ?Country),
    SUBOBJECTALLVALUESFROM(?Name, :hasTopping, _:b1),
    OBJECTUNIONOF(_:b1, ?Toppings),
    x | SUBOBJECTSOMEVALUESFROM(?Name, :hasTopping, ?Toppings) .
```

list expansion

```
?Name  $\sqsubseteq$  :NamedPizza
?Name  $\sqsubseteq$   $\exists$  hasC0. { ?Country }

?Name  $\sqsubseteq$   $\forall$  hasT. (  $\sqcup$  ?Toppings )
?Name  $\sqsubseteq$   $\exists$  hasT. ?Toppings ...
```

OTTR templates example 2

Margherita $\sqsubseteq$ NamedPizza
Margherita $\sqsubseteq \exists$ hasCountryOfOrigin.{Italy}
Margherita $\sqsubseteq \forall$ hasTopping.(Tomato $\sqcup$ Mozzarella)
Margherita $\sqsubseteq \exists$ hasTopping.Tomato
Margherita $\sqsubseteq \exists$ hasTopping.Mozzarella

Grandiosa $\sqsubseteq$ NamedPizza
Grandiosa $\sqsubseteq \forall$ hasTopping.(Tomato $\sqcup$ Jarlsberg $\sqcup$ Ham $\sqcup$ SweetPepper)
Grandiosa $\sqsubseteq \exists$ hasTopping.Tomato
Grandiosa $\sqsubseteq \exists$ hasTopping.Jarlsberg
Grandiosa $\sqsubseteq \exists$ hasTopping.Ham
Grandiosa $\sqsubseteq \exists$ hasTopping.SweetPepper

NAMEDPIZZA(Margherita, Italy, (Tomato, Mozzarella))

NAMEDPIZZA(Grandiosa, ottr:none, (Tomato, Jarlsberg, Ham, SweetPepper))

template name parameter name
 mandatory optional input
 type list input

NAMEDPIZZA(?Name : 1 class, ?Country : ? individual, ?Toppings : + class)
:: SUBCLASSOF(?Name, :NamedPizza),
 SUBOBJECTHASVALUE(?Name, :hasCountryOfOrigin, ?Country),
 SUBOBJECTALLVALUESFROM(?Name, :hasTopping, _:b1),
 OBJECTUNIONOF(_:b1, ?Toppings),
 x | SUBOBJECTSOMEVALUESFROM(?Name, :hasTopping, ?Toppings) .

list expansion

?Name $\sqsubseteq$:NamedPizza

?Name $\sqsubseteq \exists$ hasC0.{?Country}

?Name $\sqsubseteq \forall$ hasT.($\sqcup$?Toppings)

?Name $\sqsubseteq \exists$ hasT.?Toppings...

stOTTR: Easy to read and write

```
NAMEDPIZZA( ?Name : 1 class, ?Country : ? individual, ?Toppings : + class)
::  SUBCLASSOF(?Name, :NamedPizza),
    SUBOBJECTHASVALUE(?Name, :hasCountryOfOrigin, ?Country),
    SUBOBJECTALLVALUESFROM(?Name, :hasTopping, _:b1),
        OBJECTUNIONOF(_:b1, ?Toppings),
    x | SUBOBJECTSOMEVALUESFROM(?Name, :hasTopping, ?Toppings) .
```

```
NAMEDPIZZA(Margherita, Italy, (Tomato, Mozzarella))
```

```
NAMEDPIZZA(Grandiosa, ottr:none, (Tomato, Jarlsberg, Ham, SweetPepper))
```

wOTTR: OWL vocabulary for semantic web use

```
<http://draft.ottr.xyz/pizza/NamedPizza> a ottr:Template ;
ottr:hasParameter
  [ ottr:index 1 ; ottr:classVariable :pizza ] ,
  [ ottr:index 2 ; ottr:individualVariable :country;
    ottr:optional true ] ,
  [ ottr:index 3 ; ottr:listVariable (:toppings) ] .
### body:
[] ottr:templateRef t-owl-axiom:SubClassOf ;
  ottr:withValues ( :pizza p:NamedPizza ) .
[] ottr:templateRef t-owl-axiom:SubObjectHasValue ;
  ottr:withValues ( :pizza p:hasCountryOfOrigin :country ) .
[] ottr:templateRef t-owl-axiom:SubObjectAllValuesFrom ;
  ottr:withValues ( :pizza p:hasTopping _:alltoppings ) .
[] ottr:templateRef t-owl-rstr:ObjectUnionOf ;
  ottr:withValues ( _:alltoppings (:toppings) ) .
[] ottr:templateRef t-owl-axiom:SubObjectSomeValuesFrom ;
  ottr:hasArgument [ ottr:index 1; ottr:value :pizza ] ,
    [ ottr:index 2; ottr:value p:hasTopping ] ,
    [ ottr:index 3; ottr:eachValue (:toppings) ] .
```

qOTTR: query for instances of pattern

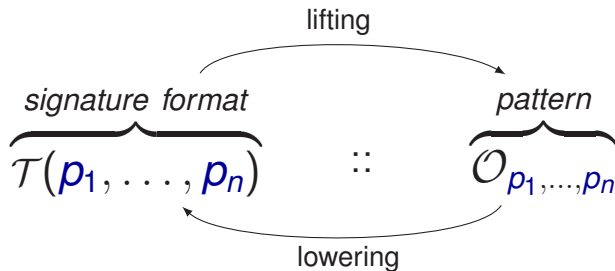
```
SELECT *
{ ?param1 rdfs:subClassOf p:NamedPizza ,
  [ owl:onProperty      p:hasTopping ;
    owl:someValuesFrom  param3item ;
    rdf:type              owl:Restriction ] ,
  [ owl:allValuesFrom   [ owl:unionOf ?param3 ;
                           rdf:type      owl:Class ] ;
    owl:onProperty      p:hasTopping ;
    rdf:type              owl:Restriction ]
OPTIONAL {
  ?param1 rdfs:subClassOf [
    owl:hasValue      ?param2 ;
    owl:onProperty     p:hasCountryOfOrigin ;
    rdf:type            owl:Restriction ]
}
?param3 (rdf:rest)* /rdf:first ?param3item
}
```

tabOTTR: tabular format for bulk template instances

9			
10	#OTTR	template	http://draft.ottr.xyz/pizza/NamedPizza
11	Pizza	Country	Toppings
12	1	2	3
13	iri	iri	iri+
14	p:Veneziana	p:Italy	p:OnionTopping p:MozzarellaTopping p:CaperTopping p:OliveTopping p:TomatoTopping p:SultanaTopping p:PineKernels
15	p:American	p:America	p:JalapenoPepperTopping p:MozzarellaTopping p:HotGreenPepperTopping p:PeperoniSausageTopping p:TomatoTopping
16	p:Margherita		p:MozzarellaTopping p:TomatoTopping
17	p:FourSeasons		p:TomatoTopping p:PeperoniSausageTopping p:CaperTopping p:MushroomTopping p:OliveTopping p:AnchoviesTopping p:MozzarellaTopping
18	p:Florentina		p:TomatoTopping p:GarlicTopping p:ParmesanTopping p:MozzarellaTopping p:OliveTopping p:SpinachTopping
19	p:PrinceCarlo		p:LeekTopping p:RosemaryTopping p:MozzarellaTopping p:TomatoTopping p:ParmesanTopping
20	p:LaReine		p:MushroomTopping p:HamTopping p:MozzarellaTopping p:OliveTopping p:TomatoTopping
21	p:American	p:America	p:TomatoTopping p:MozzarellaTopping p:PeperoniSausageTopping
22	p:Caprina		p:MozzarellaTopping p:TomatoTopping p:GoatsCheeseTopping p:SundriedTomatoTopping
23	p:PolloAdAstra		p:TomatoTopping p:RedOnionTopping p:MozzarellaTopping p:SweetPepperTopping p:CajunSpiceTopping p:GarlicTopping p:ChickenTopping
24	p:Capricciosa		p:HamTopping p:MozzarellaTopping p:OliveTopping p:TomatoTopping p:PeperonataTopping p:CaperTopping p:AnchoviesTopping
25	p:Mushroom		p:TomatoTopping p:MushroomTopping p:MozzarellaTopping
26	p:FruttiDiMare		p:MixedSeafoodTopping p:GarlicTopping p:TomatoTopping
27	p:SloppyGiuseppe		p:HotSpicedBeefTopping p:GreenPepperTopping p:OnionTopping p:MozzarellaTopping p:TomatoTopping
28	p:Cajun		p:PeperonataTopping p:TobascoPepperSauce p:TomatoTopping p:PrawnsTopping p:MozzarellaTopping p:OnionTopping
29	p:Napoletana	p:Italy	p:MozzarellaTopping p:AnchoviesTopping p:TomatoTopping p:CaperTopping p:OliveTopping
30	p:Soho		p:ParmesanTopping p:OliveTopping p:GarlicTopping p:MozzarellaTopping p:RocketTopping p:TomatoTopping
31	p:QuattroFormaggi		p:TomatoTopping p:FourCheesesTopping
32	p:Giardiniera		p:MozzarellaTopping p:TomatoTopping p:PeperonataTopping p:SlicedTomatoTopping p:MushroomTopping p:OliveTopping p:PetitPoisTopping p:LeekTopping
33	p:Siciliana		p:GarlicTopping p:HamTopping p:OliveTopping p:AnchoviesTopping p:ArtichokeTopping p:MozzarellaTopping p:TomatoTopping
34	p:Parmense		p:ParmesanTopping p:AsparagusTopping p:MozzarellaTopping p:HamTopping p:TomatoTopping
35	p:Rosa		p:GorgonzolaTopping p:TomatoTopping p:MozzarellaTopping
36			

Templates are mappings: Liftings and lowerings

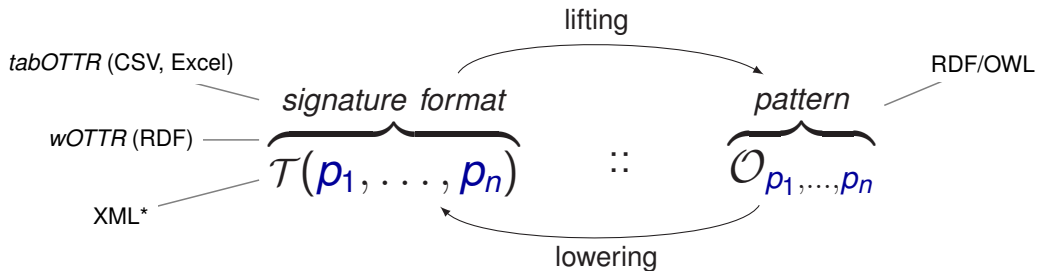
- OTTR Templates are declarative mappings/transforms
- *Generate* format and transformation specifications from a template



(* experimental)

Templates are mappings: Liftings and lowerings

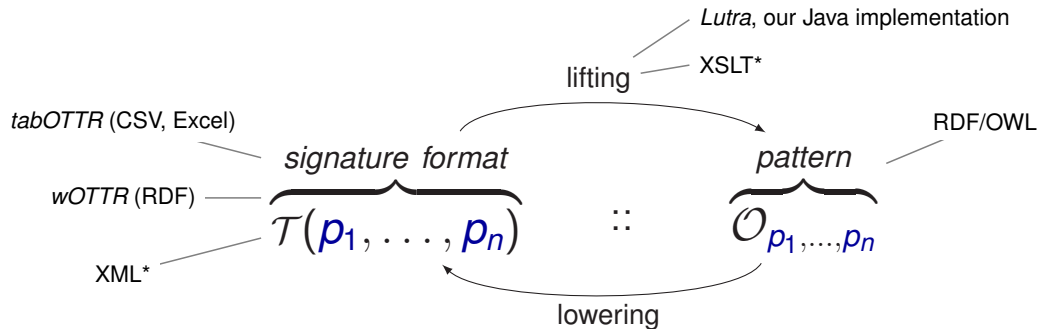
- OTTR Templates are declarative mappings/transforms
- *Generate* format and transformation specifications from a template



(* experimental)

Templates are mappings: Liftings and lowerings

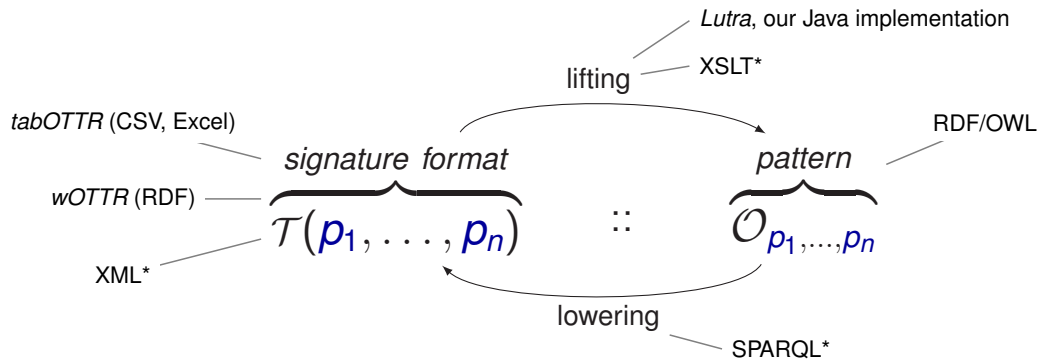
- OTTR Templates are declarative mappings/transforms
- *Generate* format and transformation specifications from a template



(* experimental)

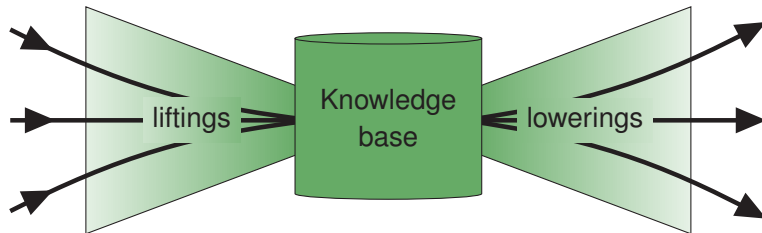
Templates are mappings: Liftings and lowerings

- OTTR Templates are declarative mappings/transforms
- *Generate* format and transformation specifications from a template



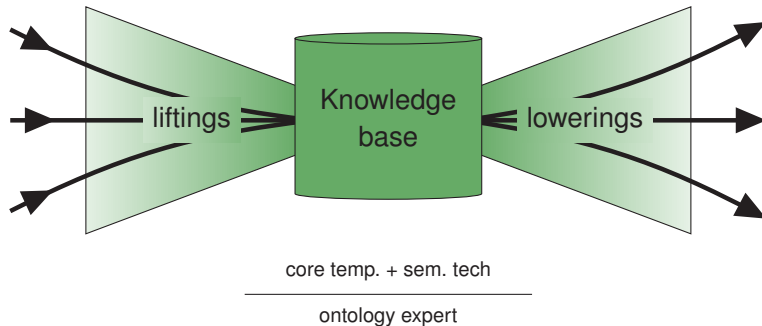
(* experimental)

Template-driven methodology



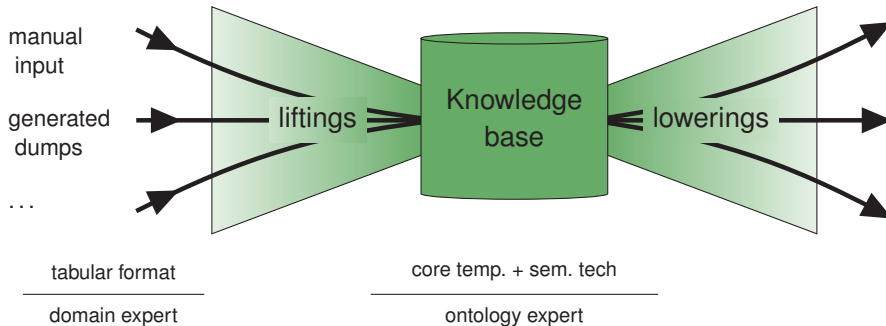
- “All” knowledge base (KB) interaction via templates
- API for RDF/OWL
- Maintain “all” KB interfaces by maintaining templates

Template-driven methodology



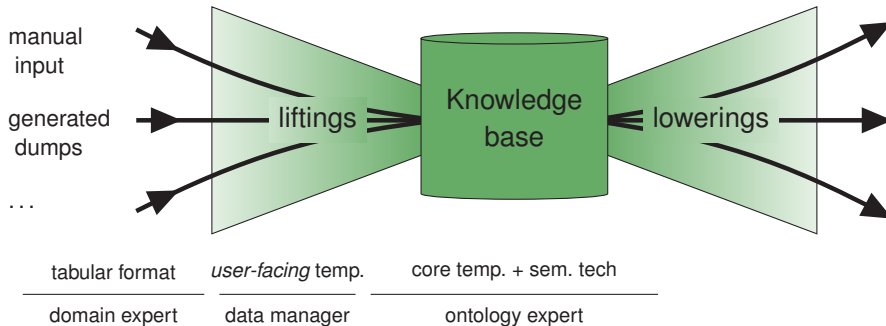
- “All” knowledge base (KB) interaction via templates
- API for RDF/OWL
- Maintain “all” KB interfaces by maintaining templates

Template-driven methodology



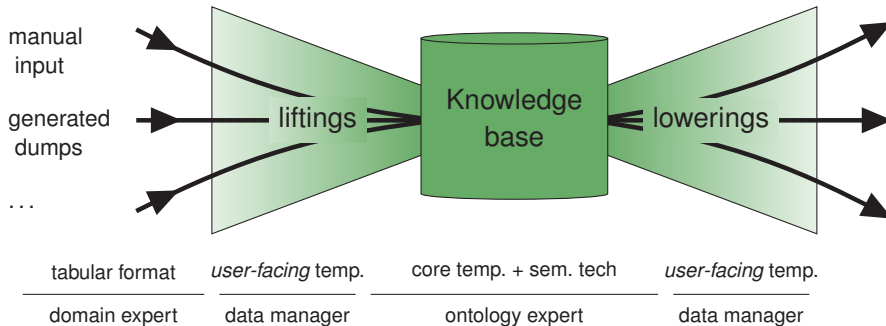
- “All” knowledge base (KB) interaction via templates
- API for RDF/OWL
- Maintain “all” KB interfaces by maintaining templates

Template-driven methodology



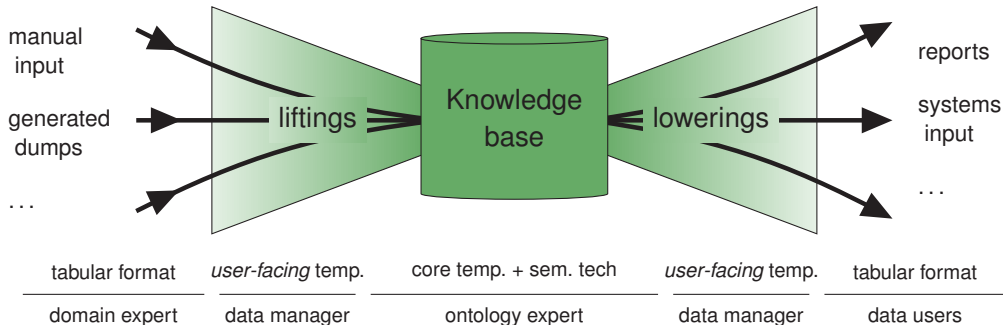
- “All” knowledge base (KB) interaction via templates
- API for RDF/OWL
- Maintain “all” KB interfaces by maintaining templates

Template-driven methodology



- “All” knowledge base (KB) interaction via templates
- API for RDF/OWL
- Maintain “all” KB interfaces by maintaining templates

Template-driven methodology



- “All” knowledge base (KB) interaction via templates
- API for RDF/OWL
- Maintain “all” KB interfaces by maintaining templates

Maintenance of template libraries: Removing redundancy

- Formal foundation of OTTR templates permits analysis
- Relations between templates, e.g., *depends*, *overlaps*, *contains*
- Two types of redundancy:

Lack of reuse: Pattern captured by existing template, refactor.

Uncaptured pattern: Pattern not captured by template, define new template and refactor

Maintenance of template libraries: Removing redundancy

- Formal foundation of OTTR templates permits analysis
- Relations between templates, e.g., *depends*, *overlaps*, *contains*
- Two types of redundancy:

Lack of reuse: Pattern captured by existing template, refactor.

Uncaptured pattern: Pattern not captured by template, define new template and refactor

$$T_1(\dots) \quad :: \quad A(\dots), B(\dots), C(\dots), D(\dots).$$
$$T_2(\dots) \quad :: \quad A(\dots), B(\dots).$$

Maintenance of template libraries: Removing redundancy

- Formal foundation of OTTR templates permits analysis
- Relations between templates, e.g., *depends*, *overlaps*, *contains*
- Two types of redundancy:

Lack of reuse: Pattern captured by existing template, refactor.

Uncaptured pattern: Pattern not captured by template, define new template and refactor

$T_1(\dots) \quad :: \quad A(\dots), B(\dots), C(\dots), D(\dots).$

$T_2(\dots) \quad :: \quad A(\dots), B(\dots).$

Maintenance of template libraries: Removing redundancy

- Formal foundation of OTTR templates permits analysis
- Relations between templates, e.g., *depends*, *overlaps*, *contains*
- Two types of redundancy:

Lack of reuse: Pattern captured by existing template, refactor.

Uncaptured pattern: Pattern not captured by template, define new template and refactor

$$T_1(\dots) \quad :: \quad T_2(\dots), C(\dots), D(\dots).$$
$$T_2(\dots) \quad :: \quad A(\dots), B(\dots).$$

Maintenance of template libraries: Removing redundancy

- Formal foundation of OTTR templates permits analysis
- Relations between templates, e.g., *depends*, *overlaps*, *contains*
- Two types of redundancy:

Lack of reuse: Pattern captured by existing template, refactor.

Uncaptured pattern: Pattern not captured by template, define new template and refactor

$$T_1(\dots) \quad :: \quad T_2(\dots), C(\dots), D(\dots).$$
$$T_2(\dots) \quad :: \quad A(\dots), B(\dots).$$
$$T_3(\dots) \quad :: \quad C(\dots), D(\dots), E(\dots).$$

Maintenance of template libraries: Removing redundancy

- Formal foundation of OTTR templates permits analysis
- Relations between templates, e.g., *depends*, *overlaps*, *contains*
- Two types of redundancy:

Lack of reuse: Pattern captured by existing template, refactor.

Uncaptured pattern: Pattern not captured by template, define new template and refactor

$$T_1(\dots) \quad :: \quad T_2(\dots), C(\dots), D(\dots).$$
$$T_2(\dots) \quad :: \quad A(\dots), B(\dots).$$
$$T_3(\dots) \quad :: \quad C(\dots), D(\dots), E(\dots).$$
$$T_4(\dots) \quad :: \quad C(\dots), D(\dots).$$

Maintenance of template libraries: Removing redundancy

- Formal foundation of OTTR templates permits analysis
- Relations between templates, e.g., *depends*, *overlaps*, *contains*
- Two types of redundancy:

Lack of reuse: Pattern captured by existing template, refactor.

Uncaptured pattern: Pattern not captured by template, define new template and refactor

$$T_1(\dots) \quad :: \quad T_2(\dots), T_4(\dots).$$
$$T_2(\dots) \quad :: \quad A(\dots), B(\dots).$$
$$T_3(\dots) \quad :: \quad T_4(\dots), E(\dots).$$
$$T_4(\dots) \quad :: \quad C(\dots), D(\dots).$$

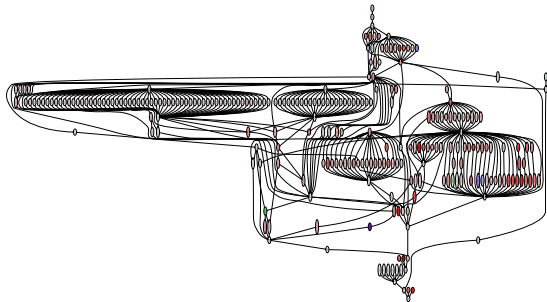
Evaluation: Aibel use case

- Aibel: EPC company



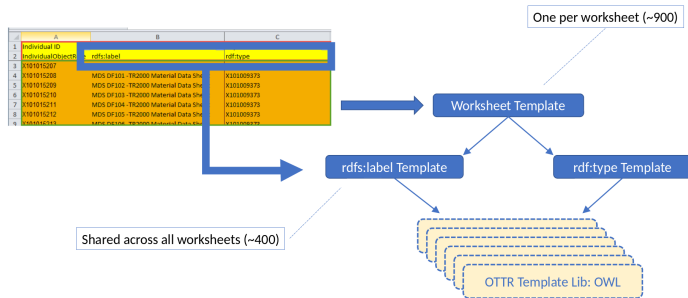
Evaluation: Aibel use case

- Aibel: EPC company
- MMD ontology:
 - 80 000 classes
 - 200 modules



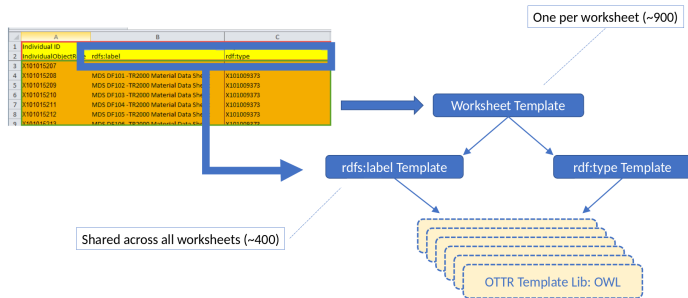
Evaluation: Aibel use case

- Aibel: EPC company
- MMD ontology:
 - 80 000 classes
 - 200 modules
- 900 spreadsheets



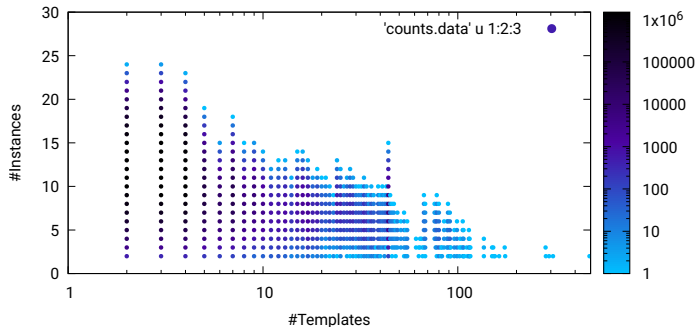
Evaluation: Aibel use case

- Aibel: EPC company
- MMD ontology:
 - 80 000 classes
 - 200 modules
- 900 spreadsheets
- All represented by OTTR templates

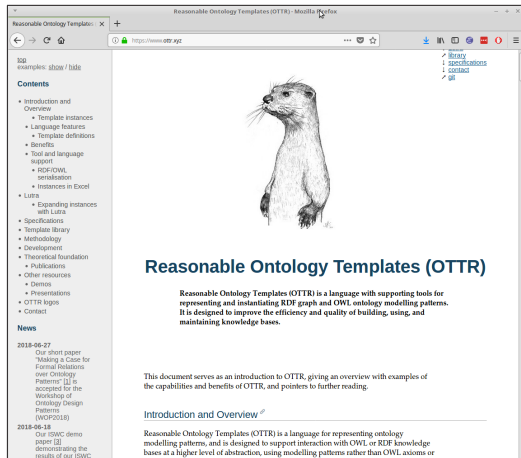


Evaluation: Aibel use case

- 900+ templates
- 550 unique templates (350 superfluous?)
- 55 million candidate redundancies
- Need heuristics for how to fix



Available at ottr.xyz



The screenshot shows a web browser window with the address bar displaying <https://www.ottr.xyz>. The page layout includes a sidebar on the left and a main content area on the right.

Sidebar:

- examples: show / hide
- Contents**
 - Introduction and Overview
 - Template instances
 - Language features
 - Template definitions
 - Benefits
 - Tool and language support
 - RDF/OWL serialisation
 - Instances in Excel
 - Lutra
 - Expanding instances with Lutra
 - Specifications
 - Template library
 - Methodology
 - Development
 - Theoretical foundation
 - Publications
 - Other resources
 - Demos
 - Presentations
 - OTTR logos
 - Contact
- News**
 - 2018-06-27
 - Our short paper "Making a Case for Formal Relations over Ontology Patterns" [1] is accepted for the Workshop of Ontology Design Patterns (WOP2018)
 - 2018-06-18
 - Our ISWC demo paper [2] demonstrating the results of our ISWC

Main Content Area:



Reasonable Ontology Templates (OTTR)

Reasonable Ontology Templates (OTTR) is a language with supporting tools for representing and instantiating RDF graph and OWL ontology modelling patterns. It is designed to improve the efficiency and quality of building, using, and maintaining knowledge bases.

This document serves as an introduction to OTTR, giving an overview with examples of the capabilities and benefits of OTTR, and pointers to further reading.

Introduction and Overview

Reasonable Ontology Templates (OTTR) is a language for representing ontology modelling patterns, and is designed to support interaction with OWL or RDF knowledge bases at a higher level of abstraction, using modelling patterns rather than OWL axioms or

Available at ottr.xyz

- *Lutra* CLI Java tool

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

## Generated by the Owl API (version 5.1.0) https://github.com/owlcs/owlapi/
./temp/ottr:s java -jar oslottr.jar -expand -in pizza.ttl

Sprefix : <http://example.com> .
Sprefix p: <http://example.com/external#> .
Sprefix owl: <http://www.w3.org/2002/07/owl#> .
Sprefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
Sprefix xml: <http://www.w3.org/XML/1998/namespace> .
Sprefix xsd: <http://www.w3.org/2001/XMLSchema#> .
Sprefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
Sbase <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
#####

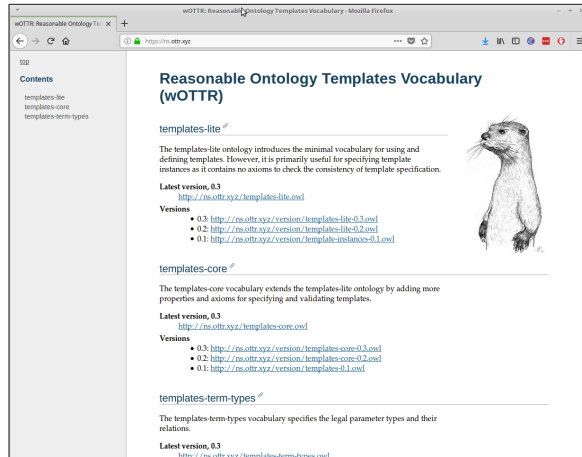
## http://example.com#AnchoviesTopping
:AnchoviesTopping rdf:type owl:Class .

## 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:onProperty p:hasTopping ;
      owl:someValuesFrom :AnchoviesTopping
    ] ,
    [ rdf:type owl:Restriction ;
      owl:onProperty p:hasTopping ;
      owl:someValuesFrom :CaperTopping
    ] ,
    [ rdf:type owl:Restriction ;
      owl:onProperty p:hasTopping ;
      owl:someValuesFrom :MozzerellaTopping
    ] ,
    [ rdf:type owl:Restriction ;
      owl:onProperty p:hasTopping ;
      owl:someValuesFrom :MushroomTopping
    ]
  .
```

Available at ottr.xyz

- *Lutra* CLI Java tool
- Specs: wOTTR, stOTTR, tabOTTR



The screenshot shows a web browser window with the address bar displaying <https://ns.ottr.xyz>. The page title is "wOTTR: Reasonable Ontology Templates Vocabulary". On the left, a sidebar contains a "Contents" section with links to "templates-lite", "templates-core", and "templates-term-types". The main content area features the heading "Reasonable Ontology Templates Vocabulary (wOTTR)". Below this, there are three sections: "templates-lite", "templates-core", and "templates-term-types". Each section includes a description of the ontology, the latest version (0.3), and a list of previous versions with their respective URLs. A small illustration of a stoat is positioned to the right of the "templates-lite" section.

Reasonable Ontology Templates Vocabulary (wOTTR)

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, 0.3
<http://ns.ottr.xyz/templates-lite.owl>

Versions

- 0.3: <http://ns.ottr.xyz/version/templates-lite-0.3.owl>
- 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, 0.3
<http://ns.ottr.xyz/templates-core.owl>

Versions

- 0.3: <http://ns.ottr.xyz/version/templates-core-0.3.owl>
- 0.2: <http://ns.ottr.xyz/version/templates-core-0.2.owl>
- 0.1: <http://ns.ottr.xyz/version/templates-0.1.owl>

templates-term-types

The templates-term-types vocabulary specifies the legal parameter types and their relations.

Latest version, 0.3
<http://ns.ottr.xyz/templates-term-types.owl>

Available at ottr.xyz

- *Lutra* CLI Java tool
- Specs: wOTTR, stOTTR, tabOTTR
- Library of OTTR templates

Reasonable Ontology Templates (OTTR) library - Mozilla Firefox

Reasonable Ontology Templates: X +

http://draft.ottr.xyz/

Available formats

- Specification
- all
- body
- equivalent
- Expansion
- all
- body
- head
- Queries
- select.body
- select.head
- string.construct
- string.update
- lowering.construct
- lowering.update
- FormatSample
- XSD format
- XML sample

Direct dependency templates

Templates instantiated in the body of this template:

- <http://candidate.ottr.xyz/owl/axiom/SubClassOf>
- <http://candidate.ottr.xyz/owl/axiom/SubObjectAllValuesFrom>
- <http://candidate.ottr.xyz/owl/axiom/SubObjectHasValue>
- <http://candidate.ottr.xyz/owl/axiom/SubObjectSomeValuesFrom>
- <http://candidate.ottr.xyz/owl/restriction/ObjectUnionOf>

Diagram of pattern

RDF graph visualisation of the expanded body:

Pattern

The pattern the template represents, i.e., the expanded template body.

Prefixes: `<http://www.cs-ode.org/ontologies/pizza/pizza.owl#>`

Available at ottr.xyz

- *Lutra* CLI Java tool
- Specs: wOTTR, stOTTR, tabOTTR
- Library of OTTR templates
- Executable demo

Reasonable Ontology Templates
Martin G. Jørgensen, Laila Harald Karlson, Daniel P. Lipp | Department of Informatics, University of Oslo

Problem

- Lack of established abstraction mechanisms for RSCOW, to capture modeling patterns
- Repetitive and tedious ontology construction
- Difficult to engage domain experts
- Low-level maintenance methods

Reasonable Ontology Templates (OTTR)

- Practical modeling language for RSCOW
- Forward reasoning, built with RSC standards
- Declarative and modular patterns
- Ensure uniform modelling
- Secure completeness of input
- Separate modelling patterns and bulk content
- Advanced template maintenance
- Available at <https://ottr.xyz>
- Open source Java implementation Lutra
- Template library, specifications, demos

Construct ontologies by instantiating OTTR templates

Multiple serialisations: stOTTR (compact), tabOTTR (tabular), wOTTR (RDF), qOTTR (SPARQL)

Ontology engineering methodology: Template abstractions adapted to different user types

Maintain ontologies by maintaining templates: Detect and remove redundancies

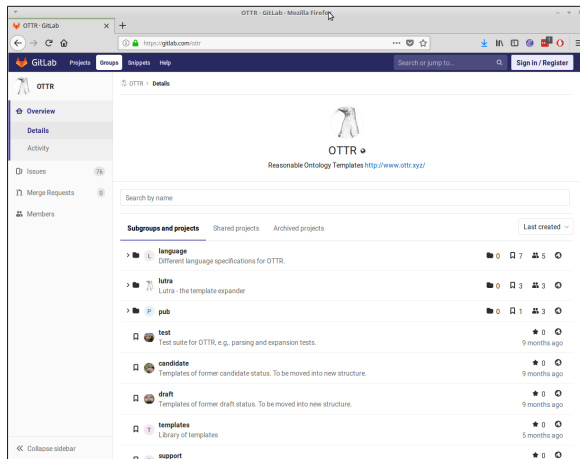
Library with independent:
• [Reasonable Ontology Templates](#) (OTTR)
• [Reasonable Ontology Templates](#) (OTTR) (Tabular)
• [Reasonable Ontology Templates](#) (OTTR) (RDF)
• [Reasonable Ontology Templates](#) (OTTR) (SPARQL)

Fast library

University of Oslo

Available at `ottr.xyz`

- *Lutra* CLI Java tool
- Specs: wOTTR, stOTTR, tabOTTR
- Library of OTTR templates
- Executable demo
- Git:
`gitlab.com/ottr/`



Summary:

- Abstraction/macro language for RDF/OWL
- Built for sem. tech use: Leverage existing W3C stack and tools
- Improved construction and maintenance of ontologies
- Methodology adapted to different users
- Library and tools available
- Industrial interest

Summary:

- Abstraction/macro language for RDF/OWL
- Built for sem. tech use: Leverage existing W3C stack and tools
- Improved construction and maintenance of ontologies
- Methodology adapted to different users
- Library and tools available
- Industrial interest

Future work:

- Editors for
 - Data manger
 - Ontology expert
- Template library maintenance
- Additional input formats:
 - Relational database
 - XML

Summary:

- Abstraction/macro language for RDF/OWL
- Built for sem. tech use: Leverage existing W3C stack and tools
- Improved construction and maintenance of ontologies
- Methodology adapted to different users
- Library and tools available
- Industrial interest

Future work:

- Editors for
Data manger
Ontology expert
- Template library maintenance
- Additional input formats:
Relational database
XML

Thanks!