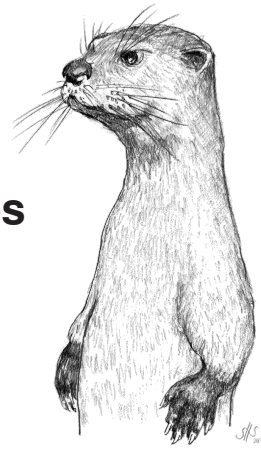
Reasonable Ontology Templates (OTTR)

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

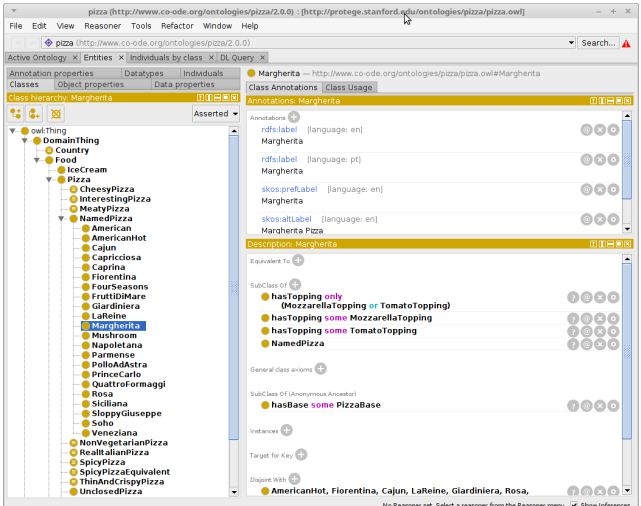
```
SubClassOf(?sub, ?super) ::
    Triple(p:Margherita, rdfs:subClassOf, p:Pizza) .

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



OTTR templates example 2

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

OTTR templates example 2

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

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

OTTR templates example 2

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

template name

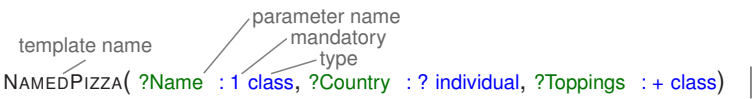
NAMEDPIZZA(?Name : 1 class, ?Country : ? individual, ?Toppings : + class) |

OTTR templates example 2

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

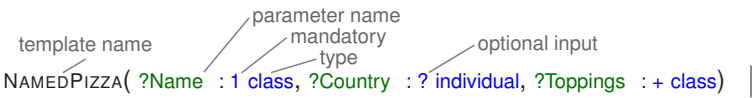
OTTR templates example 2

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



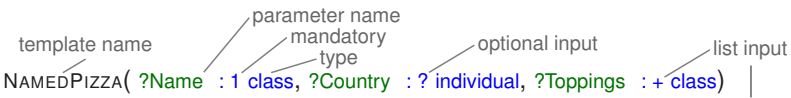
OTTR templates example 2

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



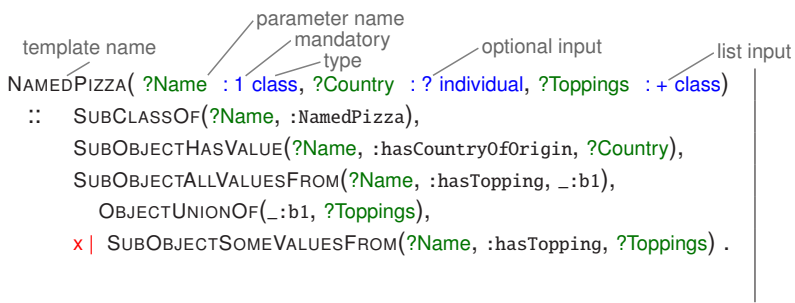
OTTR templates example 2

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



OTTR templates example 2

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



OTTR templates example 2

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

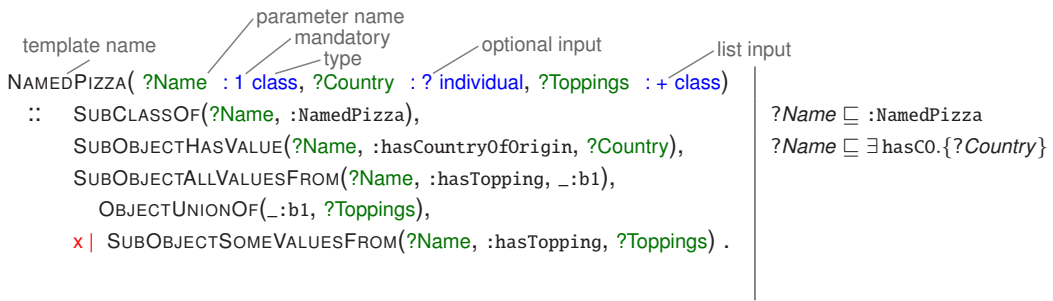
template name
parameter name
mandatory
type
optional input
list input

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

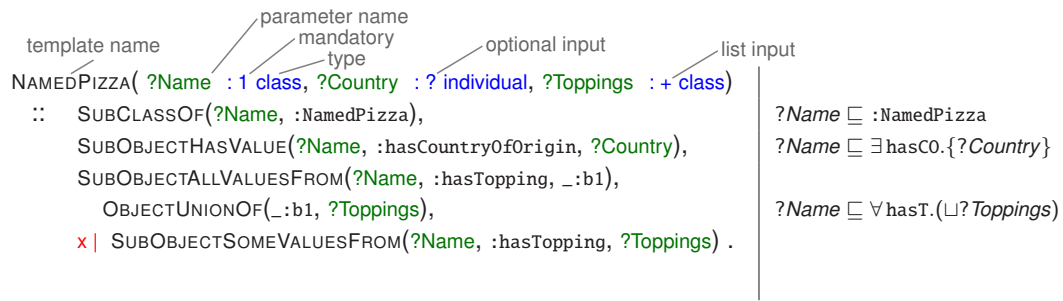
OTTR templates example 2

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



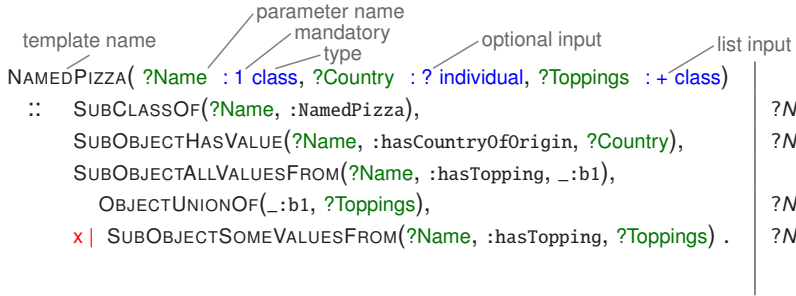
OTTR templates example 2

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



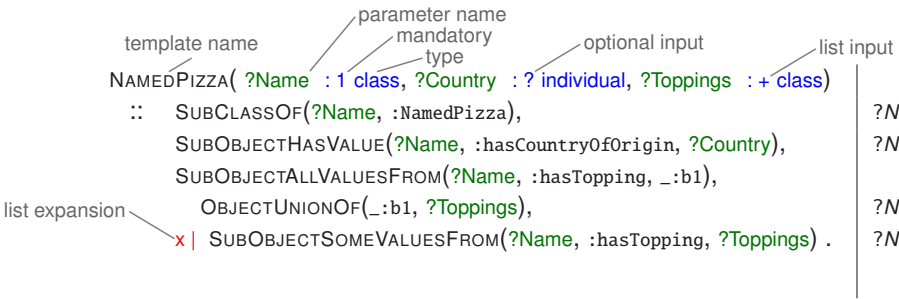
OTTR templates example 2

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



OTTR templates example 2

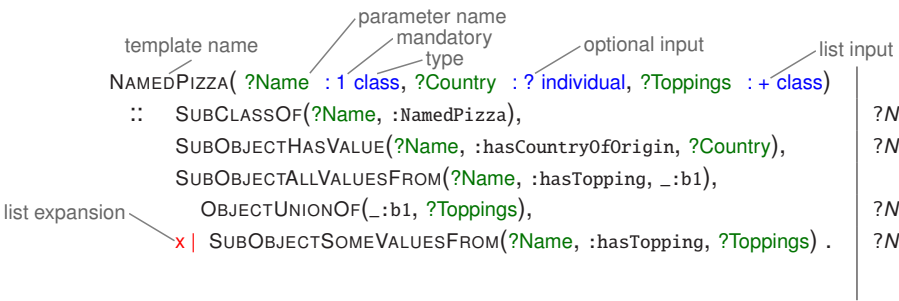
$\texttt{Margherita} \sqsubseteq \texttt{NamedPizza}$
$\texttt{Margherita} \sqsubseteq \exists\, \texttt{hasCountryOfOrigin}.\{\texttt{Italy}\}$
$\texttt{Margherita} \sqsubseteq \forall\, \texttt{hasTopping}.(\texttt{Tomato} \sqcup \texttt{Mozzarella})$
$\texttt{Margherita} \sqsubseteq \exists\, \texttt{hasTopping}.\texttt{Tomato}$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



OTTR templates example 2

```
Margherita ⊑ NamedPizza
Margherita ⊑ ∃ hasCountryOfOrigin.{Italy}
Margherita ⊑ ∀ hasTopping.(Tomato ⊔ Mozzarella)
Margherita ⊑ ∃ hasTopping.Tomato
Margherita ⊑ ∃ hasTopping.Mozzarella
```

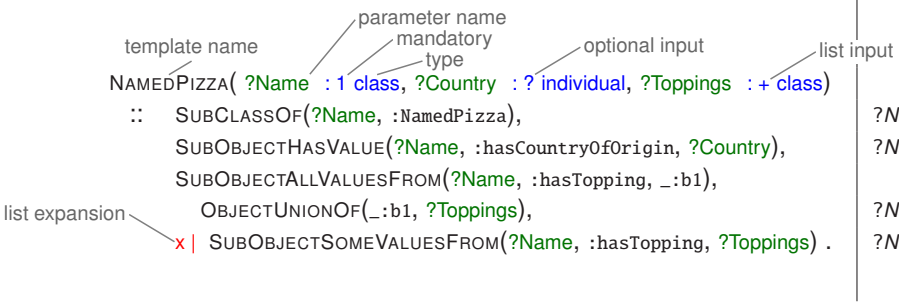
NamedPizza(Margherita, Italy, (Tomato, Mozzarella))



OTTR templates example 2

Margherita ⊑ NamedPizza
Margherita ⊑ ∃ hasCountryOfOrigin.{Italy}
Margherita ⊑ ∀ hasTopping.(Tomato ⊔ Mozzarella)
Margherita ⊑ ∃ hasTopping.Tomato
Margherita ⊑ ∃ hasTopping.Mozzarella

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



OTTR templates example 2

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

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

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),
list expansion
x | SubObjectSomeValuesFrom(?Name, :hasTopping, ?Toppings) .

$?\mathit{Name} \sqsubseteq \texttt{:NamedPizza}$
$?\mathit{Name} \sqsubseteq \exists\, \texttt{hasCO}.\{?\mathit{Country}\}$
$?\mathit{Name} \sqsubseteq \forall\, \texttt{hasT}.(\sqcup ?\mathit{Toppings})$
$?\mathit{Name} \sqsubseteq \exists\, \texttt{hasT}.?\mathit{Toppings} \ldots$

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
}
```

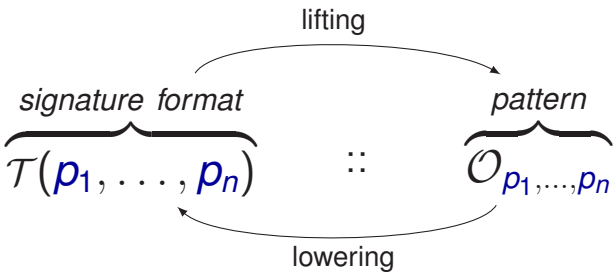
OTTR Serialisations

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:Fiorentina		p:TomatoTopping | p:GarlicTopping | p:PármesanTopping | p:MozzarellaTopping | p:OliveTopping | p:SpinachTopping
19	p:PrinceCarlo		p:LeekTopping | p:RosemaryTopping | p:MozzarellaTopping | p:TomatoTopping | p:PármesanTopping
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:PármesanTopping | 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:PármesanTopping | p:AsparagusTopping | p:MozzarellaTopping | p:HamTopping | p:TomatoTopping
35	p:Rosa		p:GorgonzolaTopping | p:TomatoTopping | p:MozzarellaTopping
36			

Templates are mappings: Liftings and lowerings

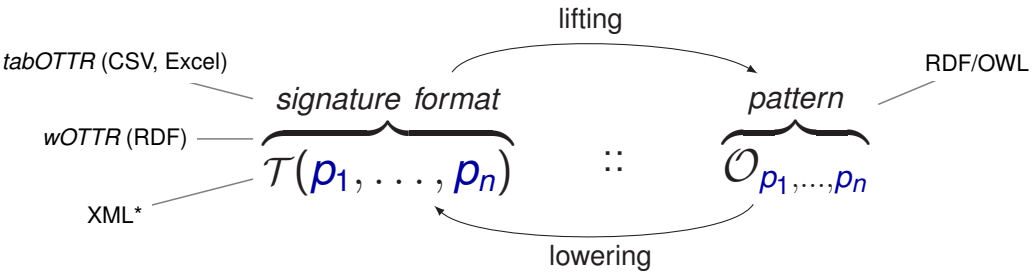
- OTTR Templates are declarative mappings/transforms
- *Generate* format and transformation specifications from a template



(* experimental)

Templates are mappings: Liftings and lowerings

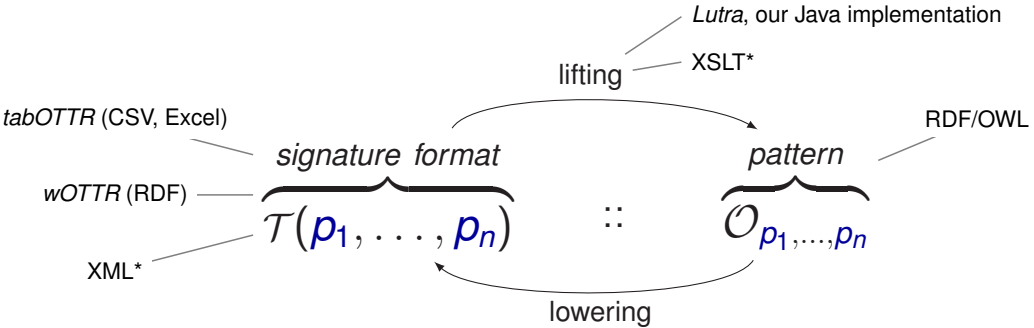
- OTTR Templates are declarative mappings/transforms
- *Generate* format and transformation specifications from a template



(* experimental)

Templates are mappings: Liftings and lowerings

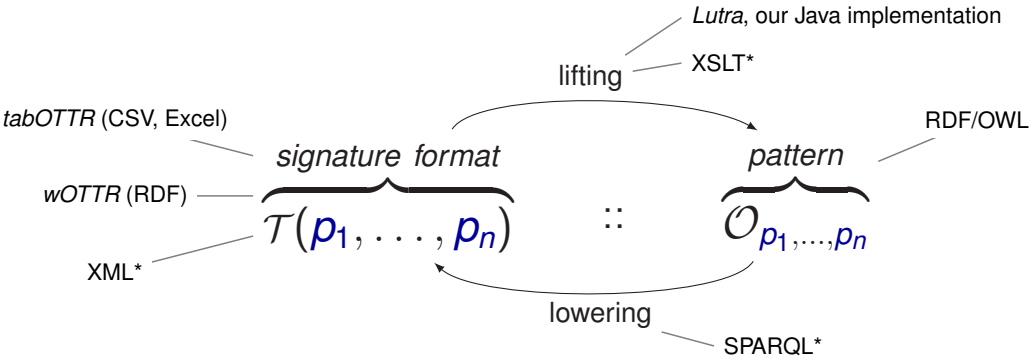
- OTTR Templates are declarative mappings/transforms
- *Generate* format and transformation specifications from a template



(* experimental)

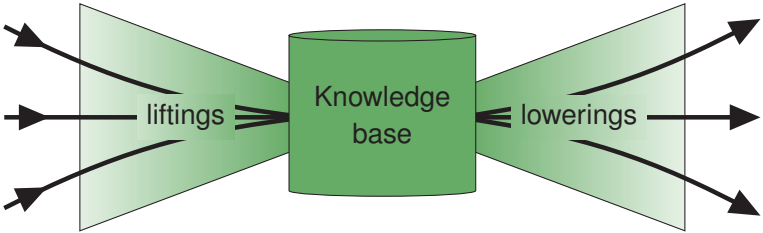
Templates are mappings: Liftings and lowerings

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



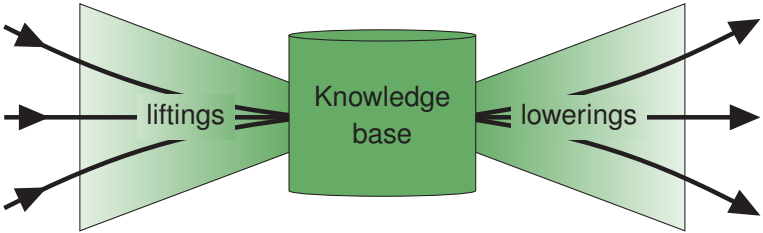
(* experimental)

Template-driven methodology



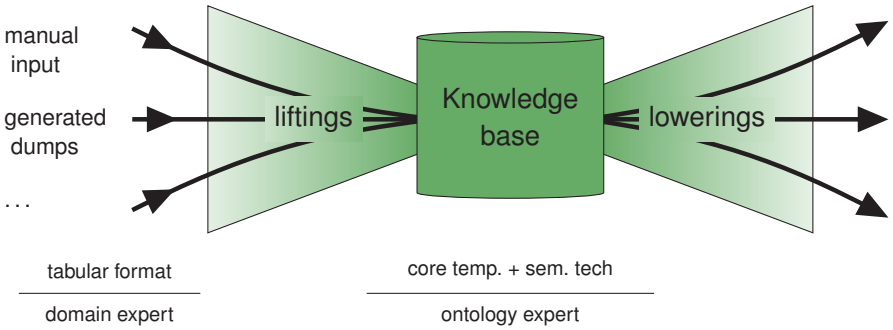
- “All” knowledge base (KB) interaction via templates
- API for RDF/OWL
- Maintain “all” KB interfaces by maintaining templates

Template-driven methodology



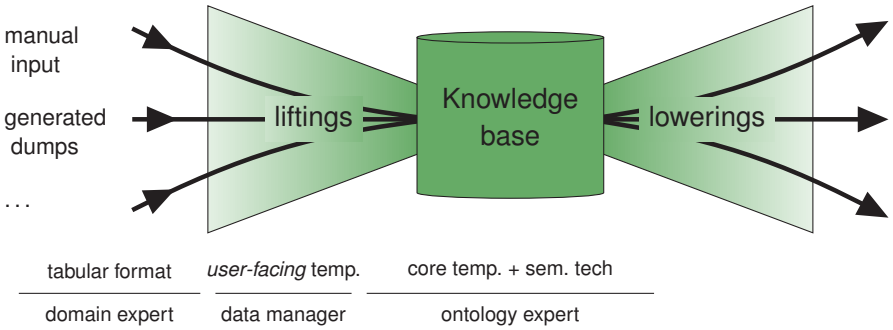
- “All” knowledge base (KB) interaction via templates
- API for RDF/OWL
- Maintain “all” KB interfaces by maintaining templates

Template-driven methodology



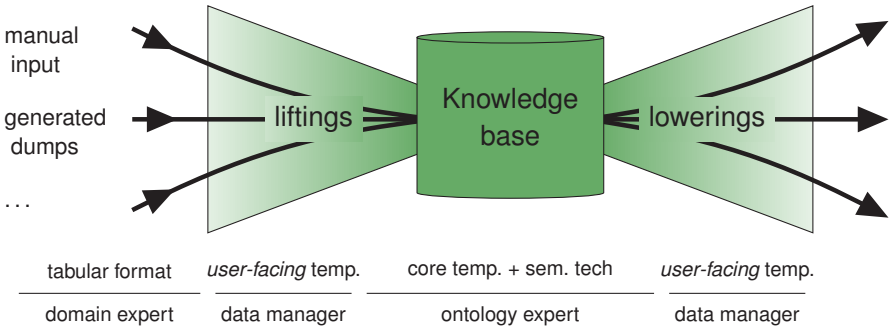
- “All” knowledge base (KB) interaction via templates
- API for RDF/OWL
- Maintain “all” KB interfaces by maintaining templates

Template-driven methodology



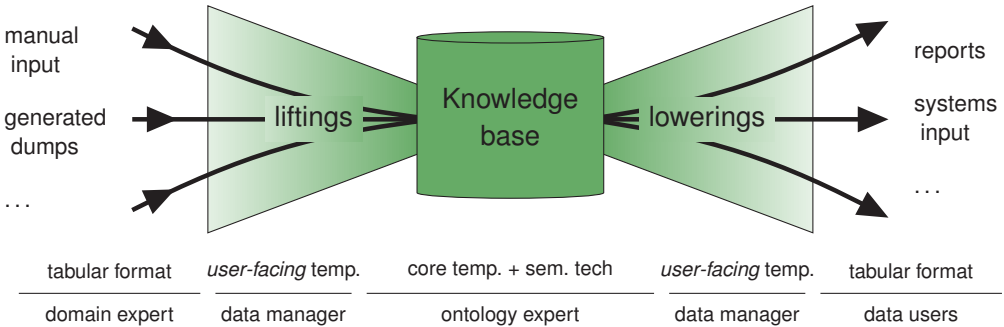
- "All" knowledge base (KB) interaction via templates
- API for RDF/OWL
- Maintain "all" KB interfaces by maintaining templates

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

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

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

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

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

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

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

$$
\begin{aligned}
T_1(\ldots) &:: T_2(\ldots), C(\ldots), D(\ldots).\\
T_2(\ldots) &:: A(\ldots), B(\ldots).\\
T_3(\ldots) &:: C(\ldots), D(\ldots), E(\ldots).
\end{aligned}
$$

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

$$
\begin{aligned}
T_1(\ldots) &:: T_2(\ldots), C(\ldots), D(\ldots).\\
T_2(\ldots) &:: A(\ldots), B(\ldots).\\
T_3(\ldots) &:: C(\ldots), D(\ldots), E(\ldots).\\
T_4(\ldots) &:: C(\ldots), D(\ldots).
\end{aligned}
$$

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

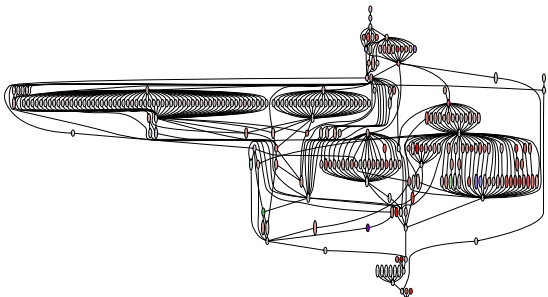
$$
\begin{aligned}
T_1(\ldots) &:: T_2(\ldots), T_4(\ldots).\\
T_2(\ldots) &:: A(\ldots), B(\ldots).\\
T_3(\ldots) &:: T_4(\ldots), E(\ldots).\\
T_4(\ldots) &:: C(\ldots), D(\ldots).
\end{aligned}
$$

Evaluation: Aibel use case

- Aibel: EPC company

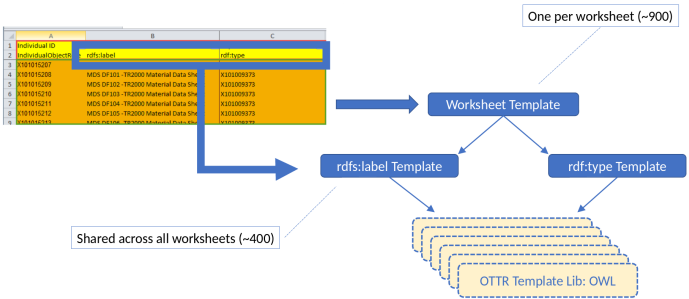
Evaluation: Aibel use case

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



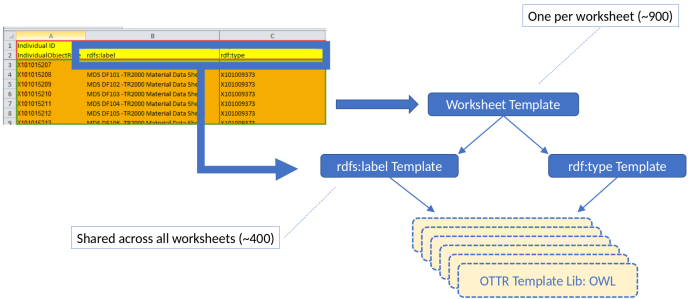
Evaluation: Aibel use case

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



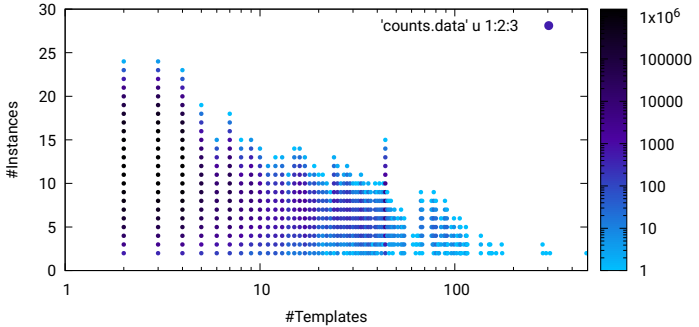
Evaluation: Aibel use case

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



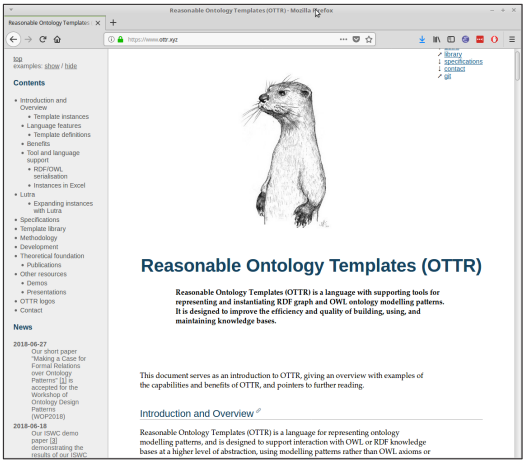
Evaluation: Aibel use case

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



Resources

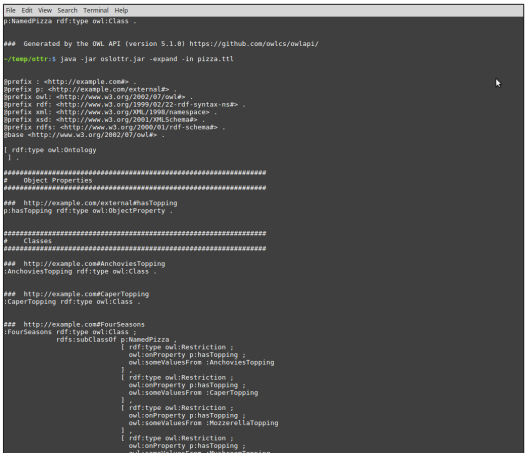
Available at `ottr.xyz`



Resources

Available at `ottr.xyz`

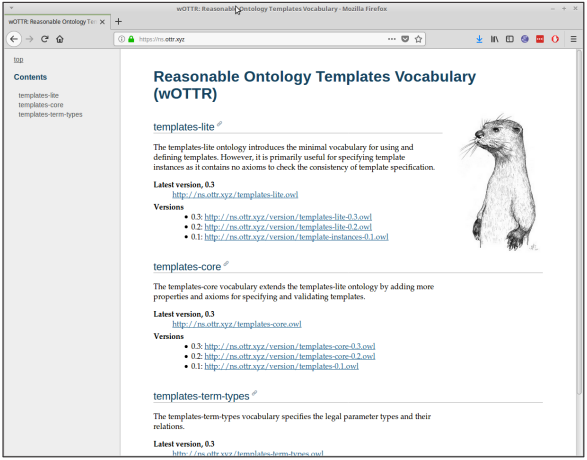
- *Lutra* CLI Java tool



Resources

Available at `ottr.xyz`

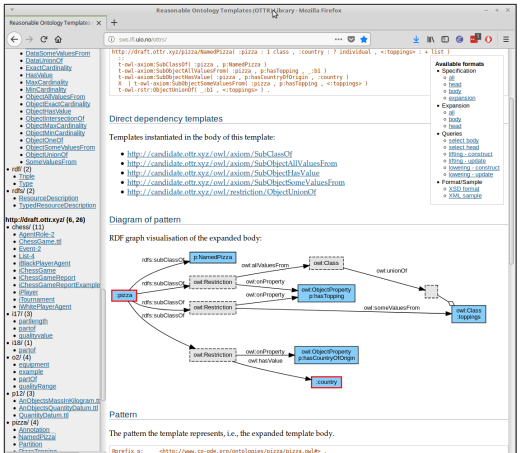
- *Lutra* CLI Java tool
- Specs: wOTTR, stOTTR, tabOTTR



Resources

Available at `ottr.xyz`

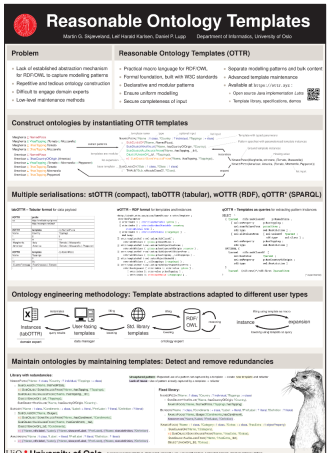
- *Lutra* CLI Java tool
- Specs: wOTTR, stOTTR, tabOTTR
- Library of OTTR templates



Resources

Available at `ottr.xyz`

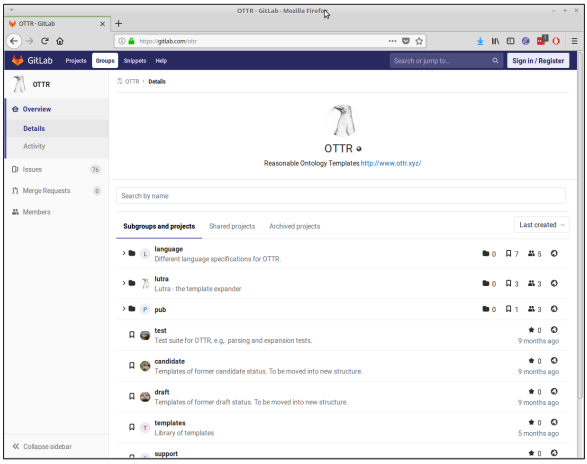
- *Lutra* CLI Java tool
- Specs: wOTTR, stOTTR, tabOTTR
- Library of OTTR templates
- Executable demo



Resources

Available at `ottr.xyz`

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



Summing up

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

Summing up

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

Summing up

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!