Software to Transform a Knowledge Graph into an Ontology

ABDOULHOUSSEN Hamza
CRESSANT Killian
ROCHU Hadrien

May 18, 2022

Contents

1. Introduction
2. State of art
   - Ontology
   - Constraints and Reasoner
   - Knowledge graph
3. From ontology to knowledge graph
   - The chess ontology
   - Visualisation on *Protégé*
   - Ontology on `python`
   - Conversion into `RDF`
   - Constraints issues
   - Triples representation
4. From knowledge graph to ontology
5. Conclusion
6. Annexe

Introduction

From this...

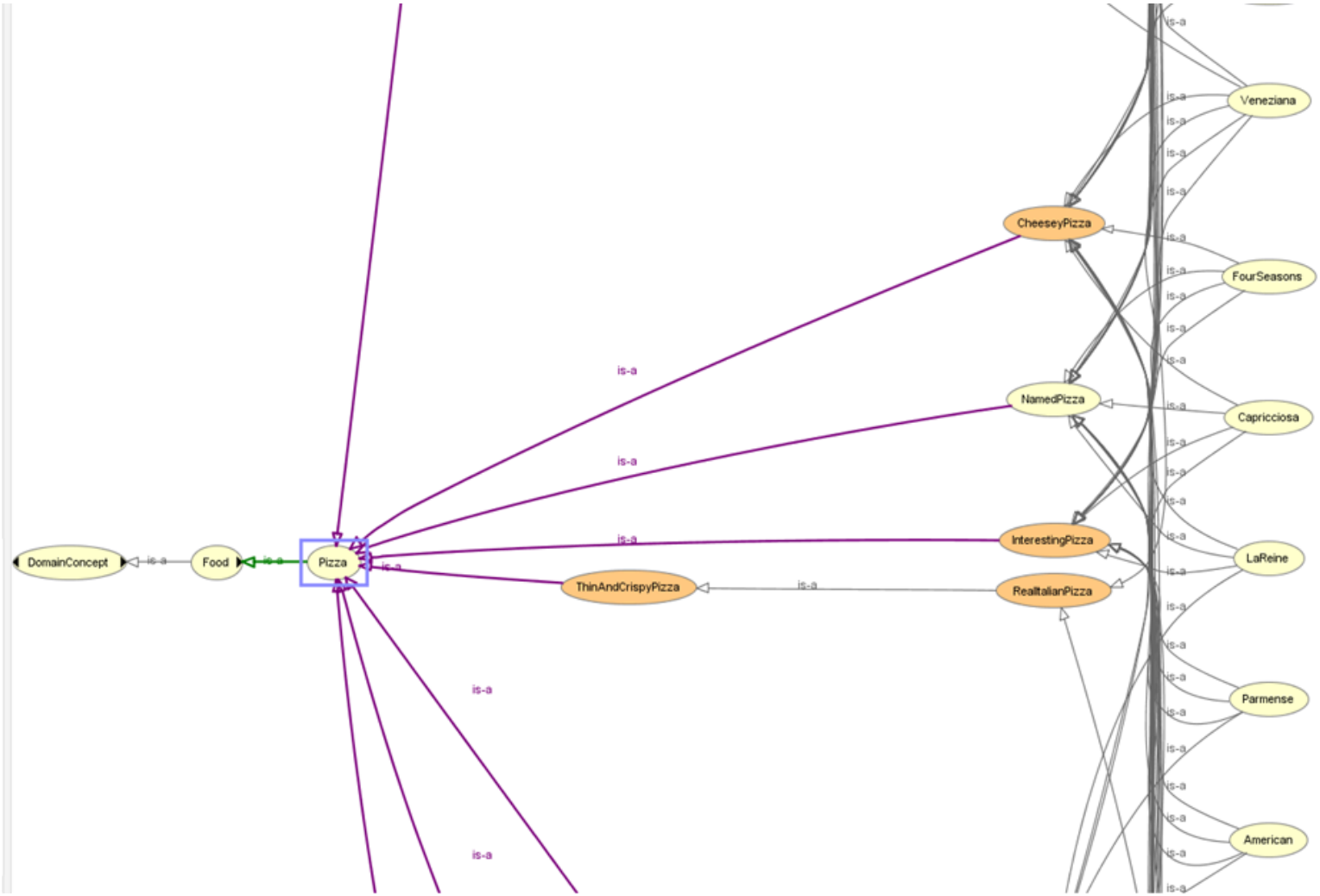


Figure: Part of Pizza ontology

...To this !

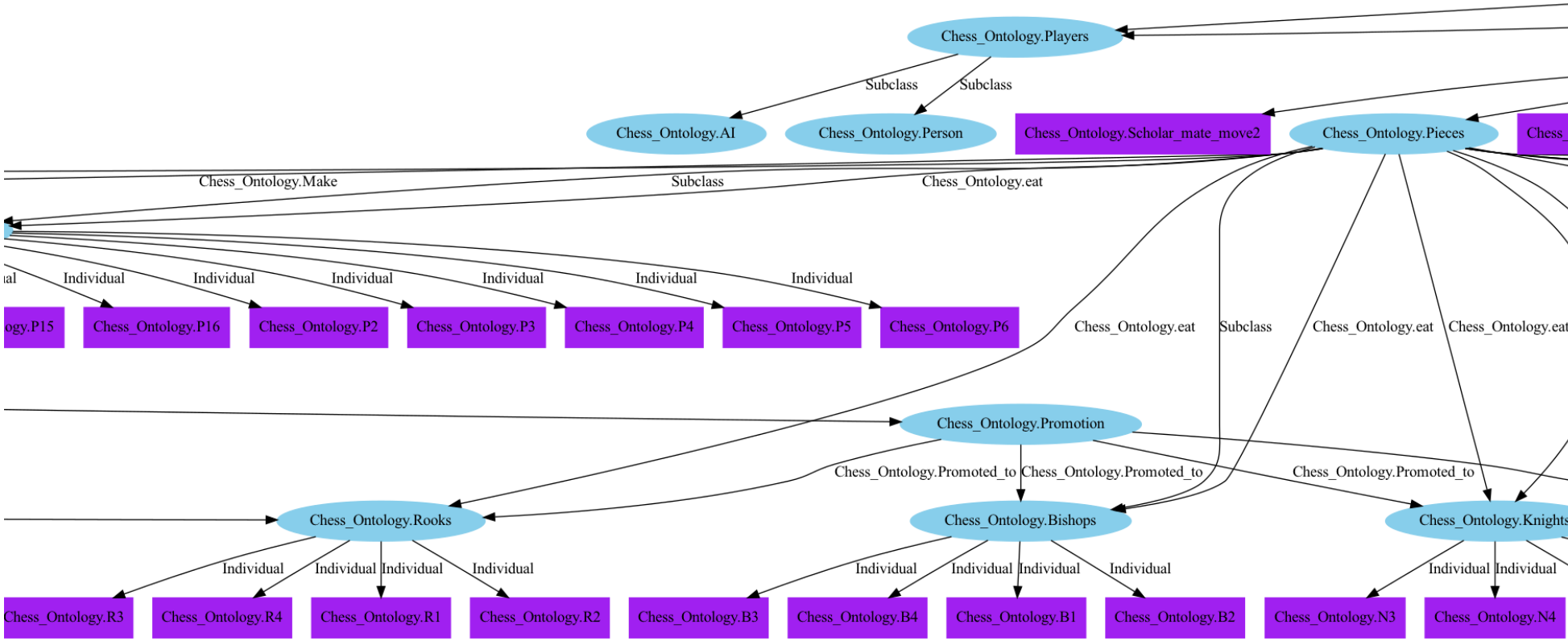


Figure: Part of a Knowledge graph of Chess game

Objectifs

- constraints modelisation

State of art

Ontology

Ontology: explicit specification of a conceptualization.

Figure: concept on pizza

Figure: concept on chess

Ontology

Ontology:

- was created by the W3C,
- manipulates knowledge,
- Initially for the web, currently in web semantics, AI, biomedical field.

$$\{\texttt{:hasTuCat} \sqsubseteq \texttt{:hasCat},\ \exists \texttt{:hasTuCat}.\top \sqsubseteq \texttt{:Turbine}\}$$

Figure: example of a small (cut) ontology O

constraints

- Constraint had been added to ontology is the last decade to make easily verification of knowledge graph and represent more information.

constraints

- Constraint had been added to ontology is the last decade to make easily verification of knowledge graph and represent more information.

$$\tau_{s_1} = \exists y(\texttt{:deplAt}(x, y)), \qquad \phi_{s_1} = (\geq_1 \texttt{:hasCat}.\top),$$
$$\tau_{s_2} = \exists y(\texttt{:hasTuCat}(x, y)), \qquad \phi_{s_2} = (\geq_1 \texttt{a.:Turbine}),$$
$$\tau_{s_3} = \texttt{:PPlant}(?x), \qquad \phi_{s_3} = (\geq_1 \texttt{:hasTurb}.s_4),$$
$$\tau_{s_4} = \texttt{:Turbine}(?x), \qquad \phi_{s_4} = (\geq_1 \texttt{:deplAt}.s_3).$$

Figure: example of constraint C

Reasoner

Reasoner

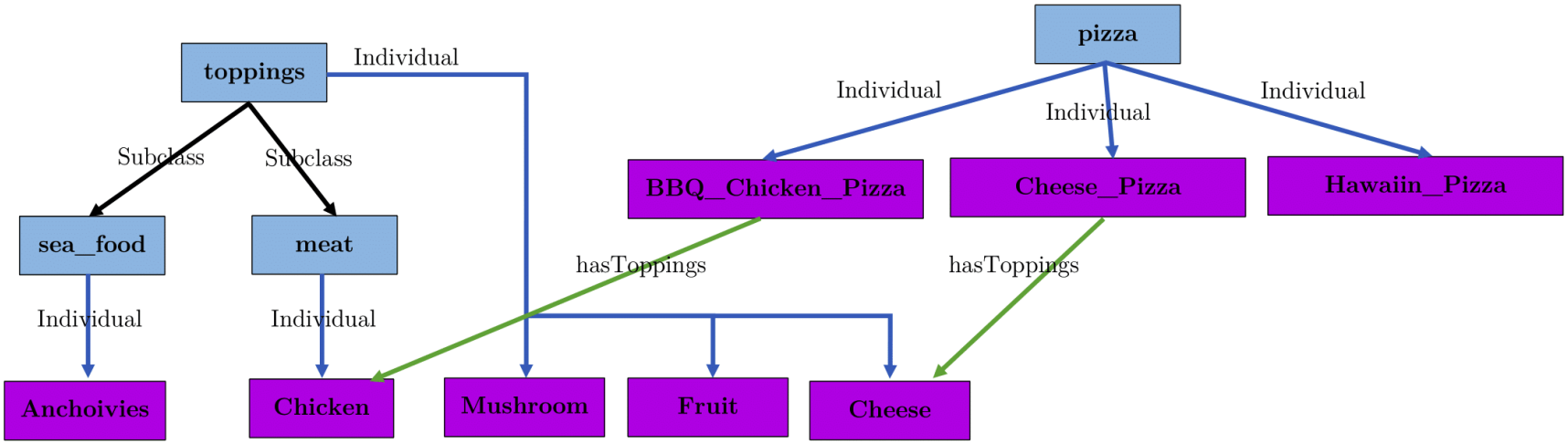


Figure: example `pizza_some`

$$\bigcup_{e \in \mathit{IndividualOf}(\mathit{meat})} \mathit{hasToppings}^{-1}(\{e\}) = \mathit{hasToppings}^{-1}(\{\mathit{Chicken}\}) = \{\mathit{BBQ\_Chicken\_Pizza}\} \quad (1)$$

Knowledge graph

Knowledge graph: a finite directed label graph that is a set of triples.
A triple is a tuple : (s, p, o) where

- s is a constant,
- p is a property,
- o is a constant or a class.

```
{(:p063, a, :PPlant),(:p063, :hasTurb, :t852),
(:t852, a, :Turbine),(:t852, :deplAt, :p063),(:t852, :hasCat,
(:t177, :deplAt, :p063),(:t177, :hasTuCat, :SGT-800),}.
```

Figure: example of a Knowledge graph G

result

$$\{s_1 \mapsto \{\texttt{:t852}, \texttt{:t177}\}, s_2 \mapsto \{\texttt{:t177}\}, s_3 \mapsto \{\texttt{:p063}\}, s_4 \mapsto \{\texttt{:t852}, \texttt{:t177}\}\}.$$

Figure: result of the previous $< O, G >$ against C

From ontology to knowledge graph

The chess ontology

Figure: Pieces

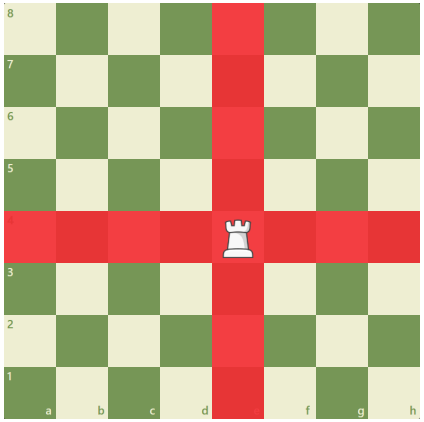


Figure: Rules and movements

The chess ontology

Figure: Pieces

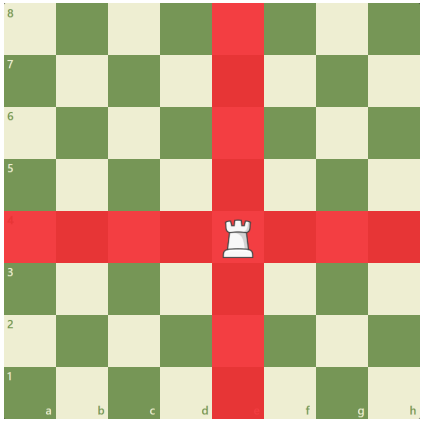


Figure: Rules and movements

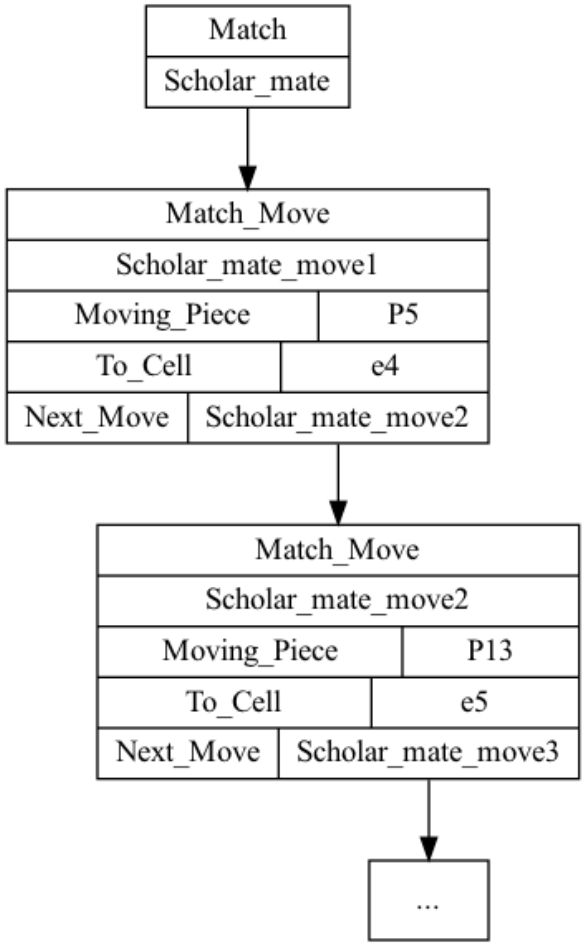


Figure: matches representation

Visualisation on *Protégé*

Plugins on *Protégé* for visualisation :

- OWLViz

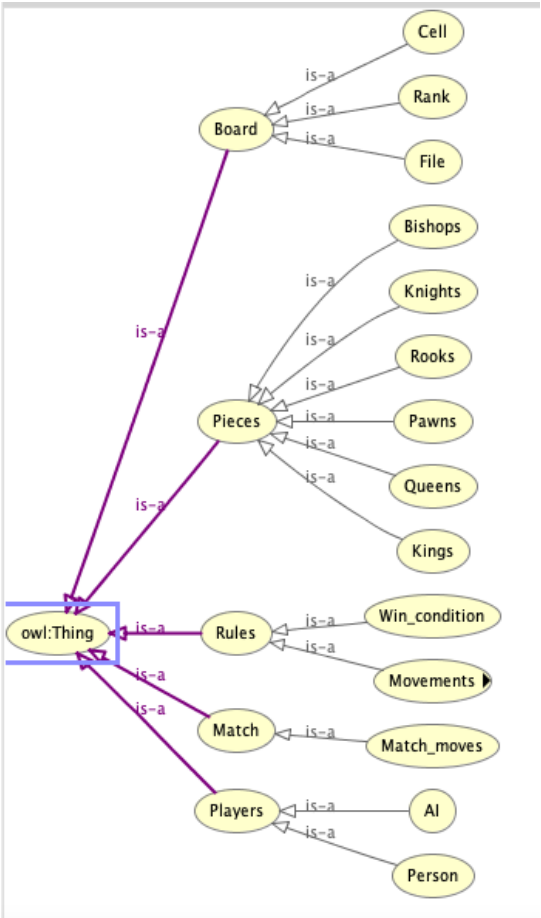
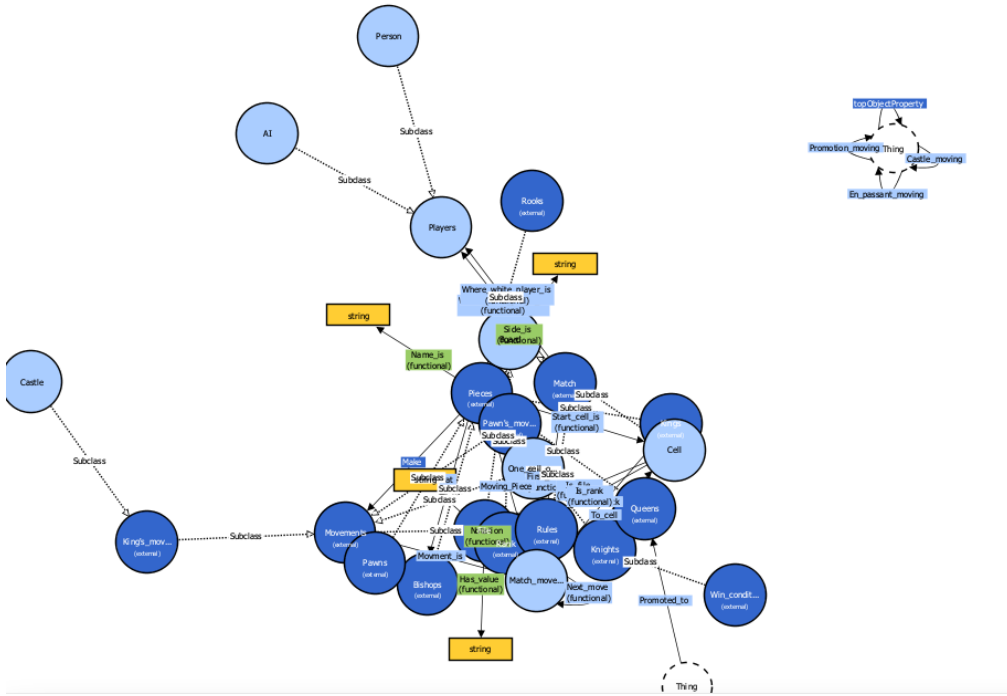


Figure: OWLViz visualisation

Visualisation on *Protégé*

- VOWL

VOWL visualisation



Unorganized VOWL visualisation

Ontology on python

Use python as object-oriented

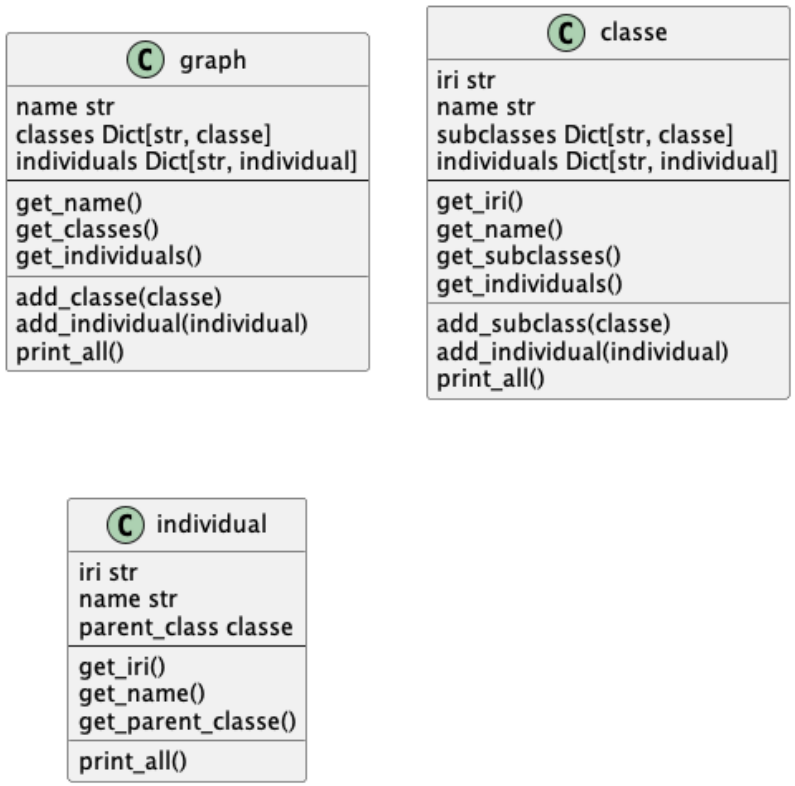


Figure: structure created on python

Ontology on python

```
    ...
Chess_Ontology.Pawns : Pawns
Chess_Ontology.Board : Board
Chess_Ontology.Rules : Rules
Chess_Ontology.Win_condition : Win_condition
Chess_Ontology.King's_movement : King's_movement
Chess_Ontology.Pawn's_movement : Pawn's_movement
Chess_Ontology.AI : AI
Chess_Ontology.Castle : Castle
Chess_Ontology.Person : Person
23 classes
```

print_all output for the chess ontology

Conversion into RDF

RDF template

```
<rdf:Description rdf:about="${subject}">
    <ex:${predicate}>
        <rdf:Description rdf:about="${object}"/>
    </ex:${predicate}>
</rdf:Description>
```

Conversion into RDF

```
1 <owl:ObjectProperty rdf:about="...#First_move">
2     <rdf:type
      ↪rdf:resource="...#FunctionalProperty"/>
3     <rdfs:domain rdf:resource="...#Match"/>
4     <rdfs:range rdf:resource="...#Match_moves"/>
5 </owl:ObjectProperty>
```

Example in OWL

Conversion into RDF

```
<owl:ObjectProperty rdf:about="...#First_move">
    <rdf:type
        ↪rdf:resource="...#FunctionalProperty"/>
    <rdfs:domain rdf:resource="...#Match"/>
    <rdfs:range rdf:resource="...#Match_moves"/>
</owl:ObjectProperty>
```

Example in OWL

```
('Chess_Ontology.Match',
    ↪'Chess_Ontology.First_move',
    ↪'Chess_Ontology.Match_moves')
```

python triple

Conversion into RDF

```
1 <rdf:Description rdf:about="Chess_Ontology.Match">
2     <ex:Chess_Ontology.First_move>
3         <rdf:Description
              ↪rdf:about="Chess_Ontology.Match_moves"/>
4     </ex:Chess_Ontology.First_move>
5 </rdf:Description>
```

output in RDF

Constraints issues

Constraints issues

Format proposed to include constraints

- For SpicyPizza

```
1    Pizza
2      and (hasTopping some
3        (PizzaTopping
4          and (hasSpiciness some Hot)))
```

Restriction in the ontology

"pizza.Pizza & pizza.hasTopping.some(pizza.PizzaTopping & pizza.hasSpiciness.some(pizza.Hot))"

The equivalent element in the RDF file

Triples representation

Triples representation from `python` using `dot` language

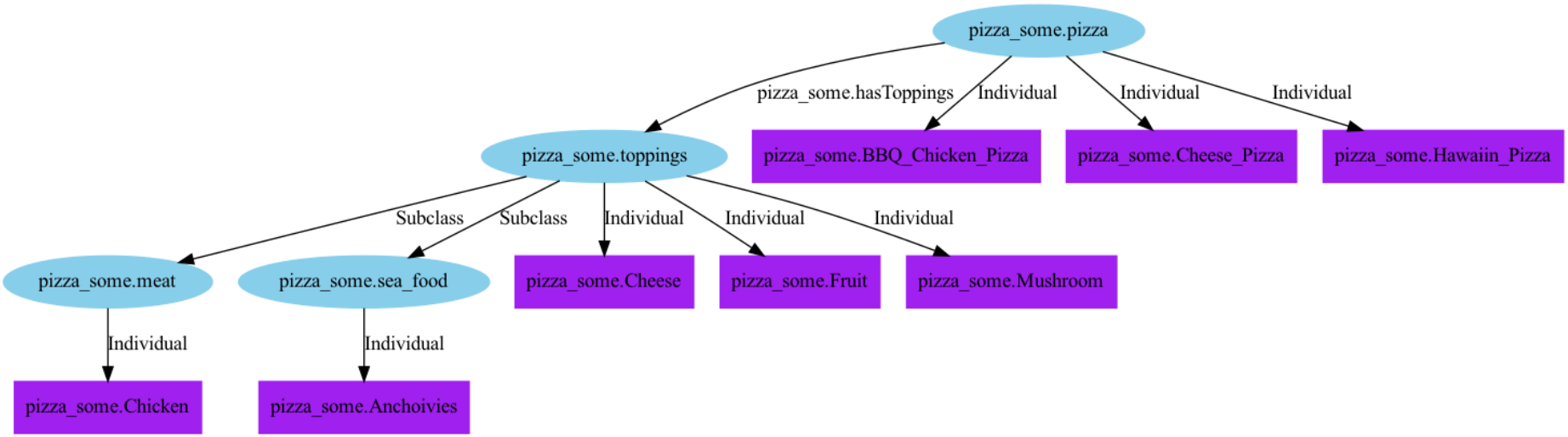


Figure: `pizza_some` representation

Triples representation

The representation is not really adapted

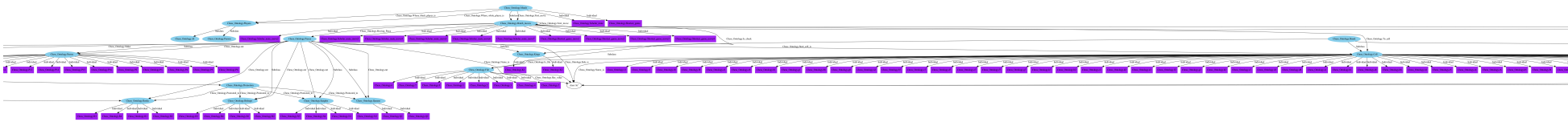


Figure: `Chess_Ontology` representation

From knowledge graph to ontology

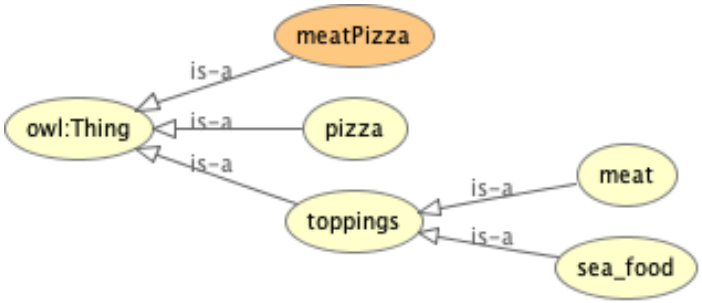


Figure: Ontology before inference

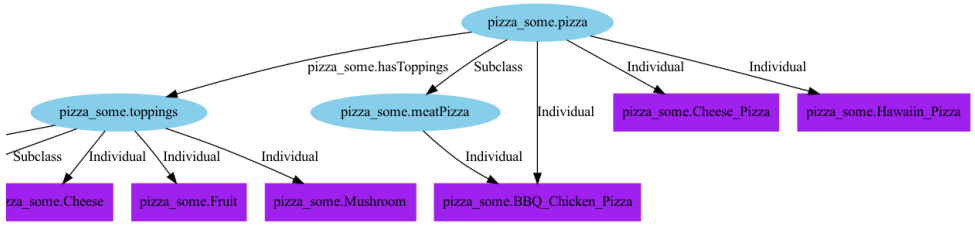


Figure: KG of pizza_some inferred (cut)

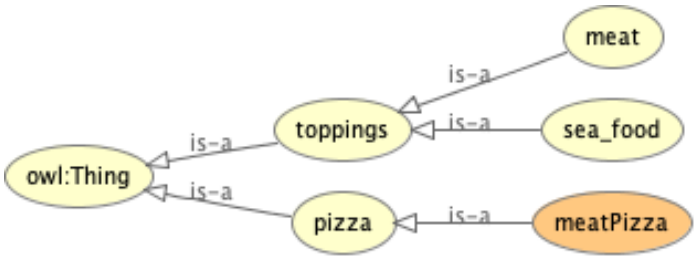


Figure: Ontology after the reasonner

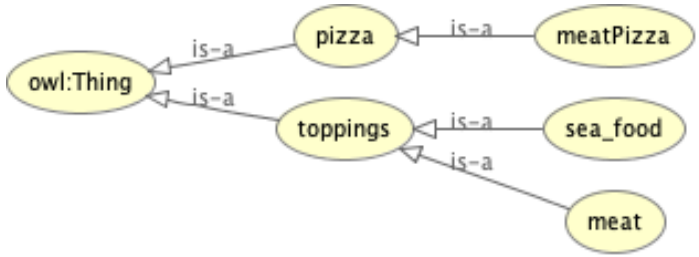


Figure: Ontology from the KG

Conclusion

Annexe

Flags on the script

```
DESCRIPTION

     -i      [required] [need argument] it is to add the owl input file path
             by default, the output is the input with '_output'
             ex : -i "resource/pizza.owl"


     -o      [need argument] add the output file path
             ex : -o "output/pizza"


     -O      [need argument] same as -O but overwrite the file if already exists
             ex : -O "output/pizza"


     -p      To print the triple added


     -s      To create standard triple without restrictions

     -r      To add reasoner before


     -kr     [need argument] To keep the ontology made after the reasoner
             the argument is the path of the new file
             ex : -kr "reasoner/pizza.owl"


     -Kr     [need argument] same as -kr but overwrite the file if already exists
             ex : -Kr "reasoner/pizza.owl"
```

Figure: Description of the script command