Software to Transform a Knowledge Graph into an Ontology

ABDOULHOUSSEN Hamza
CRESSANT Killian
ROCHU Hadrien

May 22, 2022

Contents

Introduction

Introduction - From this...

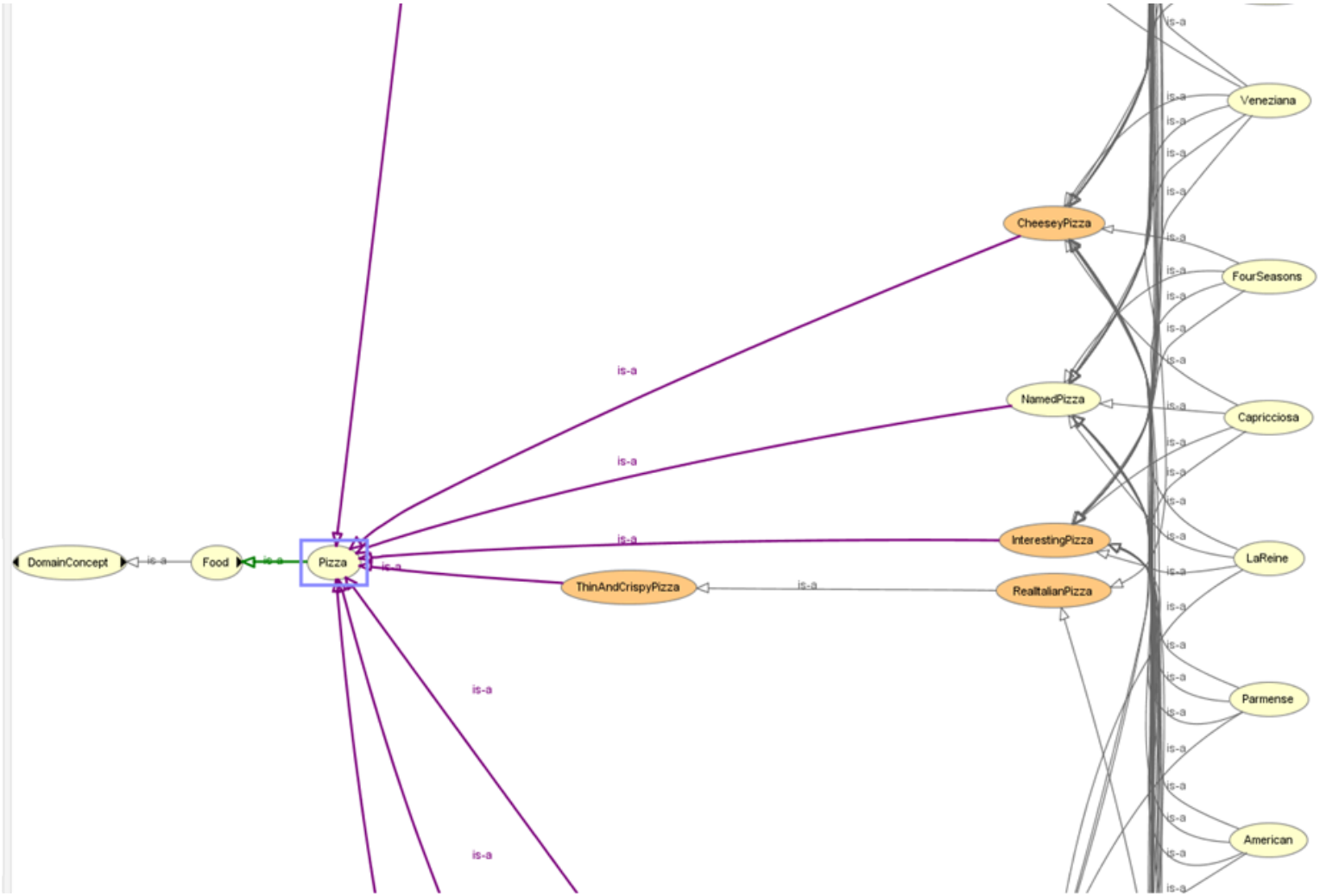


Figure: Part of Pizza ontology

Introduction - ...To this !

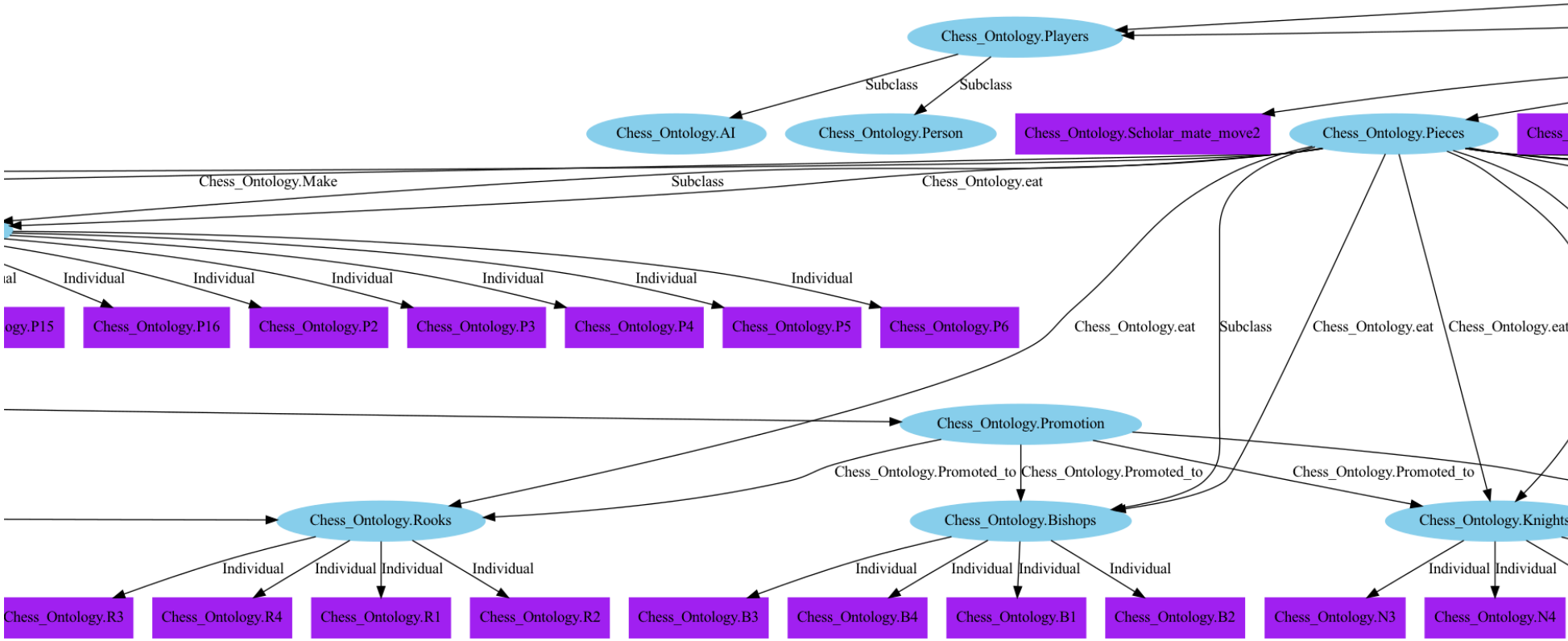


Figure: Part of a Knowledge graph of Chess game

Definitions of the concepts

Definitions of the concepts - Ontology

Definition (*Ontology*)

explicit specification of a conceptualization

Definitions of the concepts - Ontology

Definition (*Ontology*)

explicit specification of a conceptualization

Figure: concept on pizza

Figure: concept on chess

Definitions of the concepts - Ontology

Ontology:

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

Definitions of the concepts - Constraints

Definition (Constraints)

Conditions specified on classes or properties that all individuals should satisfied.

- Constraints had been added to ontologies in the last decade to make easily verification of knowledge graph and represent more information.

Example

- and
- or
- not
- some
- only
- value
- min
- max
- exactly

Definitions of the concepts - Reasoner

Definition (Reasoner)

A semantic reasoner is a piece of software
able to infer logical consequences from a set of asserted facts or axioms.

Definitions of the concepts - Reasoner

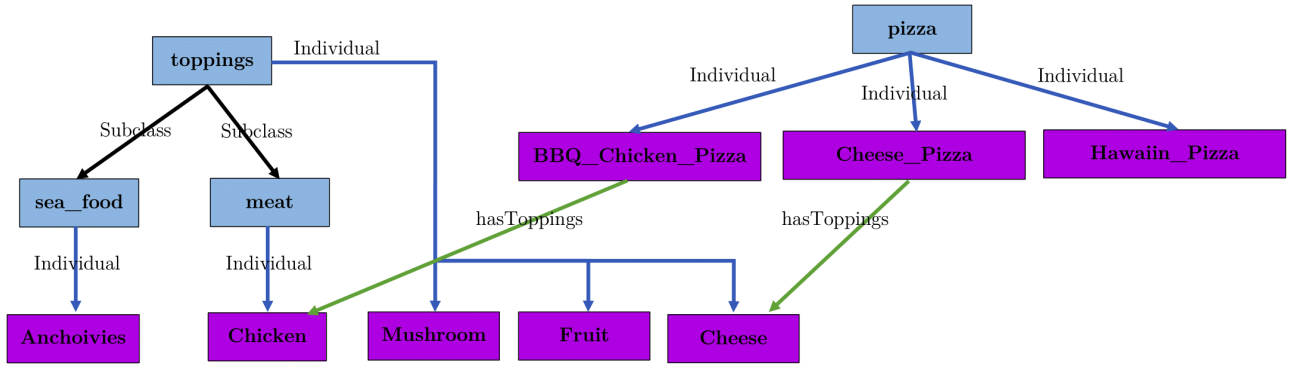


Figure: example `pizza_some`

hasTopping some meat

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

Definitions of the concepts - Knowledge graph

Definition (*Knowledge graph*)

A *knowledge graph* is 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 [1]

From ontology to knowledge graph (KG)

From ontology to KG - The chess ontology

Figure: Pieces

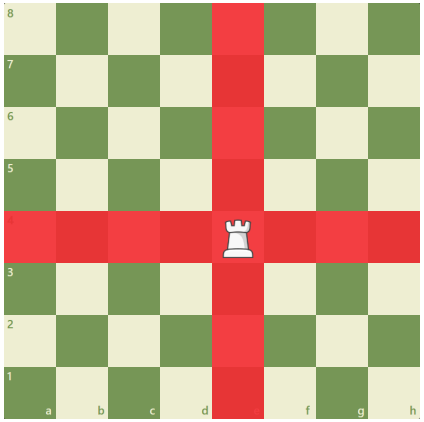


Figure: Rules and movements [2]

From ontology to KG - The chess ontology

Figure: Pieces

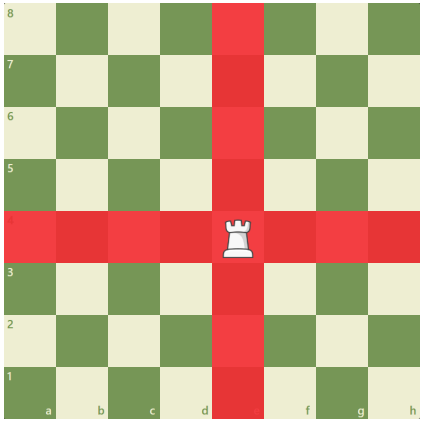


Figure: Rules and movements [2]

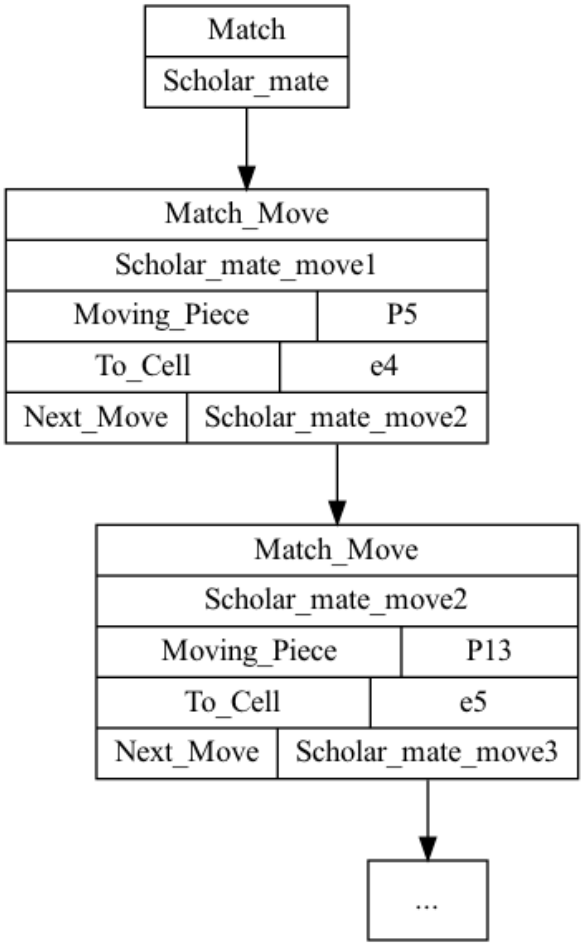


Figure: matches representation

From ontology to KG - Visualisation on *Protégé*

Plugins on *Protégé* for visualisation :

- OWLViz

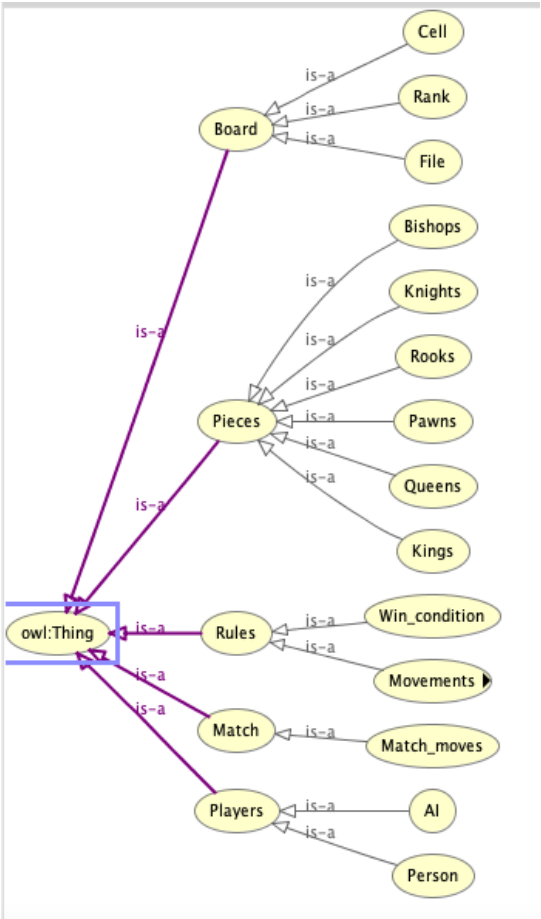
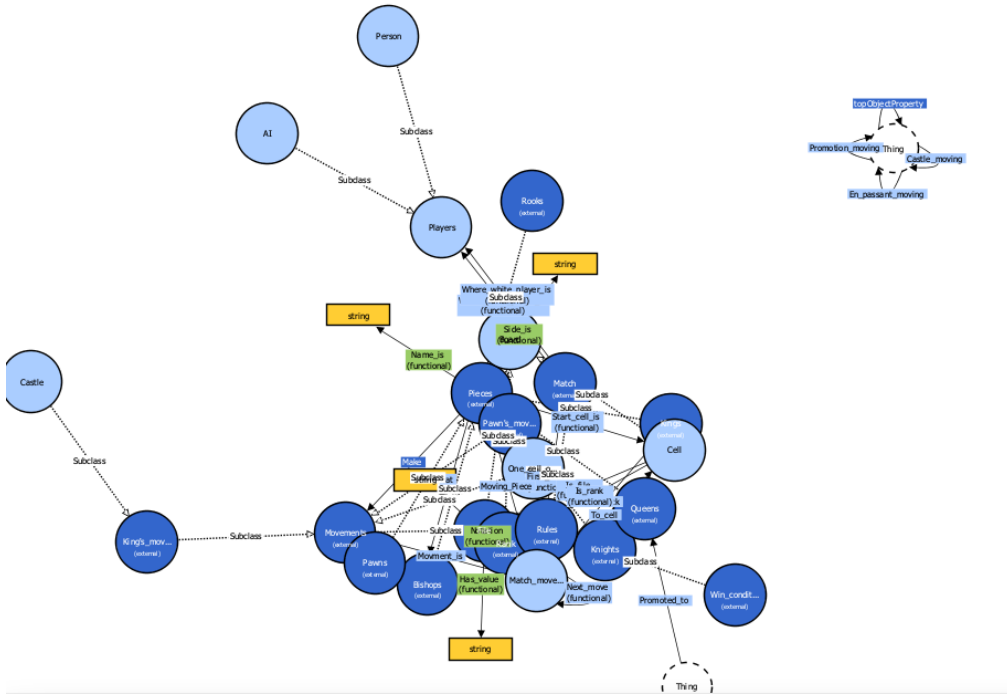


Figure: OWLViz visualisation

From ontology to KG - Visualisation on *Protégé*

- VOWL

VOWL visualisation



Unorganized VOWL visualisation

From ontology to KG - Ontology on python

Use python as object-oriented

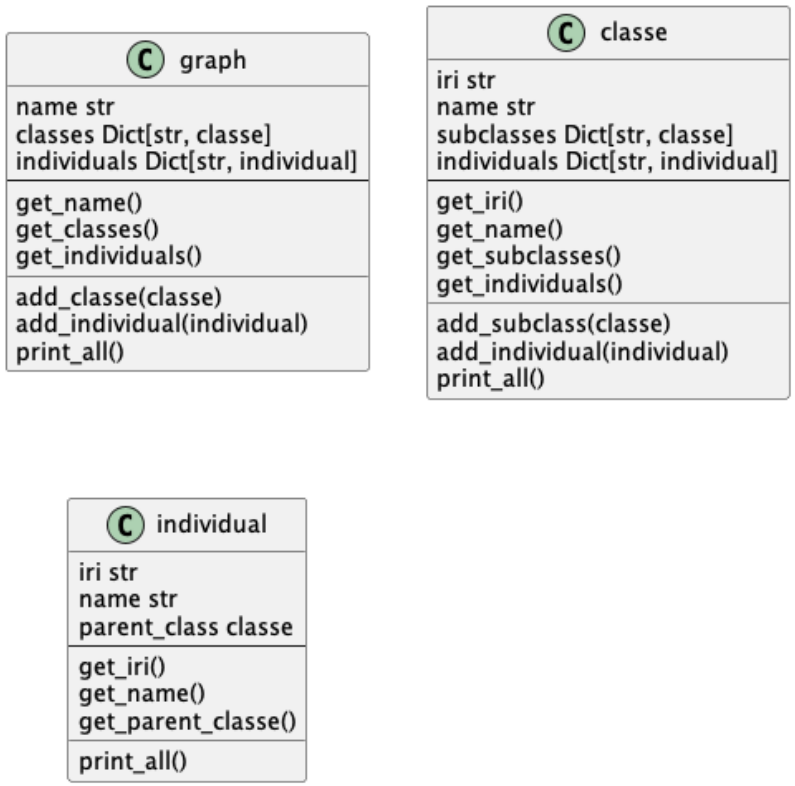


Figure: structure created on python

From ontology to KG - Ontology on python

```
1  ...
2  Chess_Ontology.Pawns : Pawns
3  Chess_Ontology.Board : Board
4  Chess_Ontology.Rules : Rules
5  Chess_Ontology.AI : AI
6  Chess_Ontology.Castle : Castle
7  Chess_Ontology.Person : Person
8  23 classes
```

print_all output for the chess ontology

Class count	23
Object property count	18
Data property count	4
Individual count	137

Figure: The elements in *protégé*

From ontology to KG - Conversion into RDF

RDF template [3]

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

From ontology to KG - 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

From ontology to KG - 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

From ontology to KG - 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

From ontology to KG - 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

From ontology to KG - Triples representation

Triples representation from `python` using `dot` language

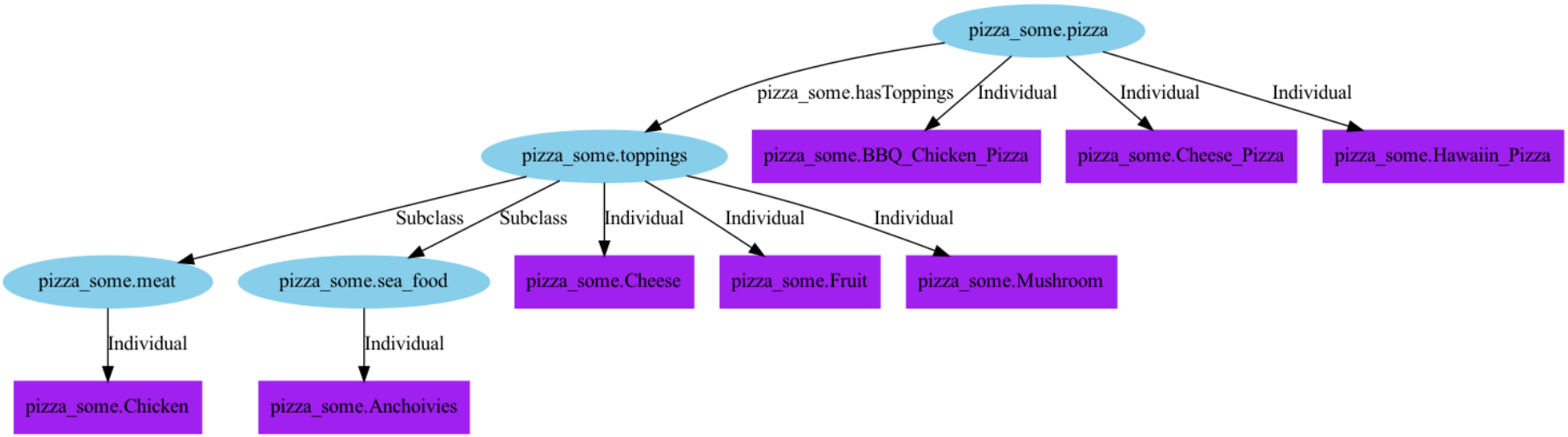


Figure: `pizza_some` representation

From ontology to KG - Triples representation

The representation is not really adapted

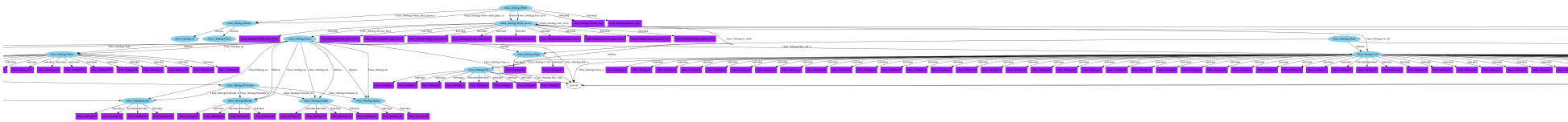


Figure: `Chess_Ontology` representation

From knowledge graph to ontology

From KG to ontology - Graphics

We keep the reasoner inferences [4]

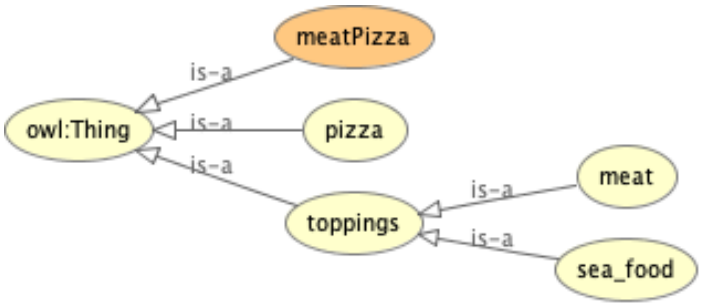


Figure: Ontology before inference

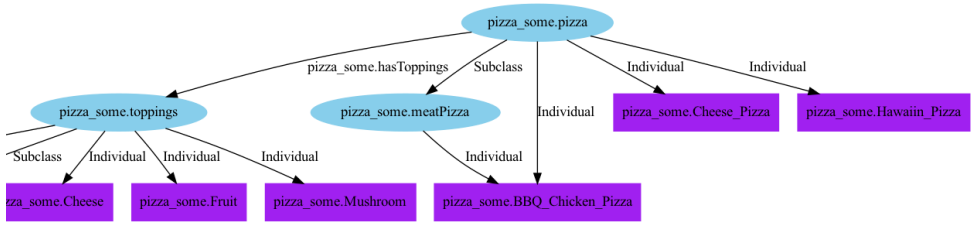


Figure: KG of pizza_some inferred (cut)

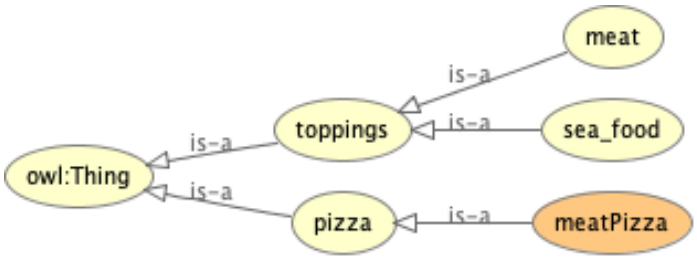


Figure: Ontology after the reasonner

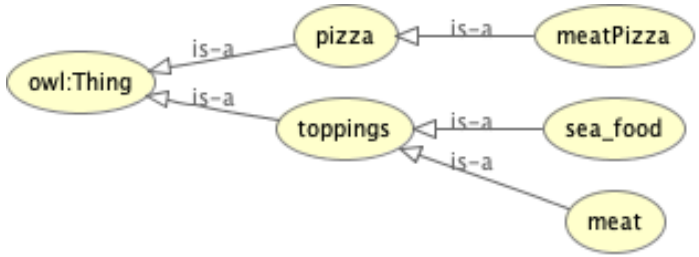


Figure: Ontology from the KG

Conclusion

Conclusion - Work done

Work done

- Learn about ontologies, knowledge graphs, reasoner
- familiarise with ontologies and *Protégé*
- Partial conversion from ontology to KG
- Conversion from our KG to ontology
- Add visualisation tools

However

- Restrictions issues are not completely solved
- The visualisations are really limited

Conclusion - What we learned

We learned

- the concepts behind ontologies, KG, reasoner
- to use *Protégé*
- to use python as Object-Oriented language
- to use the `owlready2` package
- the dot language and the `graphviz` package to make visual graphs
- to make clean code and documentation
- to propose flags in scripts and commands
- to make tests (on github)

Bibliography

[1] Ognjen Savkovic, Evgeny Kharlamov, Steffen Lamparter, "SHACLE constraint Validation over Ontology-enhanced KGs via Rewriting,", December 2018, https://www.inf.unibz.it/krdb/KRDB%20files/tech-reports/KRDB18-03.pdf.

[2] Adila Krisnadhi, Pascal Hitzler, *Modeling With Ontology Design Patterns: Chess Games As a Worked Example*, https://people.cs.ksu.edu/~hitzler/pub2/01-chess-example.pdf.

[3] Computer Science and Operations Research Montreal University, *Triples in RDF/XML*, https://www.iro.umontreal.ca/~lapalme/ForestInsteadOfTheTrees/HTML/ch07s01.html, April 2019.

[4] Bart Bogaerts, Maxime Jakubowski, Jan Van den Bussche, *SHACL: A Description Logic in Disguise*, https://arxiv.org/pdf/2108.06096.pdf#section.A.1, 2021.

Appendix

Appendix - 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

Appendix - Our repisotory on github

Github repository

`https://github.com/Hamza-ABDOULHOUSSEN/PIDR_knowledge_graph`