Transformations on Knowledge Representation between OWL and RDF Knowledge Graphs: a Study Case

Mariano Ferreirone[1,2,3], Mario Lezoche[1], Diego Torres[2,3], Herv´e Panetto[1]
[1] University of Lorraine, CNRS, CRAN, F54000 Nancy, France
mariano-julian.ferreirone@univ-lorraine.fr, mario.lezoche@univ-lorraine.fr,
herve.panetto@univ-lorraine.fr
[2] LIFIA, CICPBA-Facultad de Informatica, UNLP, Argentina
mferreirone@lifia.info.unlp.edu.ar, dtorres@lifia.info.unlp.edu.ar
[3] Depto. CyT, UNQ, Bernal, Argentina
dtorres@unq.edu.ar, mferreirone@unq.edu.ar

Abstract. Is well known that the semantic web is having a tremendous impact on many aspects of the world and that it's a wave that is far away from going down. Ontology and Knowledge graphs are two methods of knowledge representation that are part of the basis of this wave, and both have their pros and cons. In this work, there is an analysis over the relation between a rigid schema knowledge representation as OWL, and a simple and more flexible one like RDF. This is based on the attempt of transforming an OWL knowledge graph into an RDF knowledge graph, taking into account the interesting possibility of combining knowledge graphs that were created with different levels of schema formalization. The work also presents a case of study on the chess domain.

Keywords: Ontology · Knowledge graph · OWL · RDF · knowledge representation

1 Introduction

During the last years, Knowledge Graphs (KG) have been demonstrated that are well known alternatives for knowledge discovering. There are well known cases in the industry that uses KG in order to organize their data and then discover underlying knowledge that are not directly present in their data sources [8].

A KG is a formal representation of knowledge in the form of a labeled directed graph, where the nodes represent concepts or an actual entity from the real world, meanwhile the edges represent different relations between these nodes [5]. A standard data model to represent KG is the Resource Description Framework (RDF), [10] which use triples of the form (subject, predicate, object). Additionally, the RDF model is complemented by query languages in which SPARQL is the most prominent.

KGs could be represented with different levels of schema formalization which co-exists. The simplest one is the aforementioned RDF, which is a data model and it doesn't support semantics itself. It's based on XML, so it only inherits the XML datatype definitions. Another level is RDFS (Resource Description Framework Schema) which is an extension of RDF, and it introduces simple constraints and semantics, as class and property subtypes, property range and domain restrictions. Finally, OWL (Web Ontology Language) that introduces

This paper is partially supported by funding provided by the STIC AmSud program, Project 22STIC-01.

Received April 2023 Accepted July 2023; Published August 2023
https://doi.org/10.24215/15146774e019

several ontological characteristics on top of RDFS [1]. Taking into account the aforementioned, the authors found interesting to investigate how these different KGs, with different levels of schema formalization, could be combined, in order to increase the interoperability between them.

During the last years, the amount of RDF knowledge graphs has been increasing and the most big and popular knowledge representations of this type are created with this formalization language. As an example, Wikidata [2] is constructed on RDF and it has several billions of triples [4].

The RDF knowledge graphs could have a schema behind in order to add consistency to the model and the data, and this is the case when they are ontology-based. In the case that the KG is purely developed in RDF, the lack of a formal schema does not ensure the consistency of the data, and there's a lack of semantic expressiveness in comparison with an ontology. Another strong point to highlight is that the knowledge graphs which use a shared ontology are more inter-operable since the ontology structure is unambiguous and it has an accepted and common meaning in the community [5]. On the other side, RDF/S knowledge graphs are more flexible with the addition of new information, and since they don't have to do a strict review of the structure, is more efficient from a computational point of view. On the other hand, OWL KG has a lot of expressiveness in their semantics, but they are not as fast computationally as the Knowledge graphs [7]. These are reasons that make attractive the idea of working in a bidirectional connection or transformation of these two knowledge representation models.

Having an ontology with all the instances from the equivalent knowledge graph is useful for visualizing the hierarchy and the structure in a clear way, and it also ensures the consistency of the model. Furthermore, having into account that RDF Knowledge graphs are likely to have a good efficiency, but they lack on the consistency since they don't have a natural semantic schema structure behind, there's a gap that awakes the scientific research interest, in order to add this expressiveness that, for example, OWL has, creating a good combination of both characteristics.

The approach that the authors of this article have developed, in order to start a solution path for the aforementioned points of improvement, is to create a set of rules or steps with the objective of transforming an OWL Knowledge graph into an RDF knowledge graph, with the intention of analyzing what is lost in the middle and define further steps.

There are some existent interesting works which are related to converting an RDF and OWL into different formats. An example of this is the converter developed as part of the tool named CoGui[3] by the GraphiK team at LIRMM [3] . In this case, the transformation is done from RDF to conceptual graphs, and the resulting OWL file is exported to different languages. When referring

[1] Wikidata, OWL, https://en.wikipedia.org/wiki/Web$_{O}ntology_{L}anguage$

[2] Wikidata, https://www.wikidata.org/wiki/Wikidata:Main$_{P}age$

[3] CoGui Homepage, https://www.lirmm.fr/cogui/3/index.html. Last accessed 5/5/2022

Received April 2023 Accepted July 2023; Published August 2023

to RDF, currently, OWL rules, constraints and type dis-junctions are ignored. Another interesting work is the converter tool of the University of Manchester [4]. This converter doesn't have the possibility of converting an OWL structure to a simple RDF syntax. Another interesting concept to take into account is the ontology alignment or ontology matching, which is based on generating a set of correspondences between concepts, properties or instances of different structured KGs, with the objective of unifying them into a new one [2].

This work introduced the initial analysis of applying transformation rules which transform an OWL KG into an RDF KG. The approach is conducted over a study case of a chess ontology. Learned lessons and challenges are reported.

The article is structured as follows: in section 2 there's a general description about the transformation process that the authors have designed and applied. In section 3 we describe the creation of an OWL ontology over the chess domain, and the application of the transformation to the corresponding file. Finally, in section 4 the learned lessons are described, and the next steps and further challenges are mentioned.

2 Transformation

The transformation developed in this work consists of the creation of a program, with the objective of converting an OWL KG into different types of RDF triples representations, which means, RDF Knowledge graphs.

As a first step, the user can make the decision of working with the original OWL KG input, or if it's desired to infer the file in order to include also the implicit axioms in the forward steps, using a reasoner engine and creating a new version of the file. On the second step, the program reads the chosen input KG and obtains all the elements from it, including classes, sub-classes, properties, etc., and assigns each of them to a graph structure that was previously created by the authors. This graph structure has the following classes: Graph, which contains the name of the graph, all the classes and all the individuals; Class, that contains information related to a class, as his name and iri, all the subclasses and the individuals of the class; Individual, which contains the name, iri and parent class of the individual. The figure 1 describes the graph structure. Once this graph is fed, the data and the knowledge are ready for being processed.

As the next step, the generation of an intermediate output, which is going to be structured as simple triple stores in the form (subject, predicate, object) takes place. Then, the triple stores are translated to RDF/XML syntax, including also those triples that are formed by OWL elements and don't correspond to the RDF semantic level. This is done using a template, which is described in the figure 2, and creating a customized RDF structure, where some triples could be compounded, concatenating information in the predicate or in the object. Finally, the latter generated output will pass through a cleaner with the objective of creating an RDF/XML output in the corresponding RDF semantic level.

[4] OWL Syntax Converter, University of Manchester http://mowl-power.cs.man.ac.uk:8080/converter/. Last accessed 5/5/2022

Received April 2023 Accepted July 2023; Published August 2023

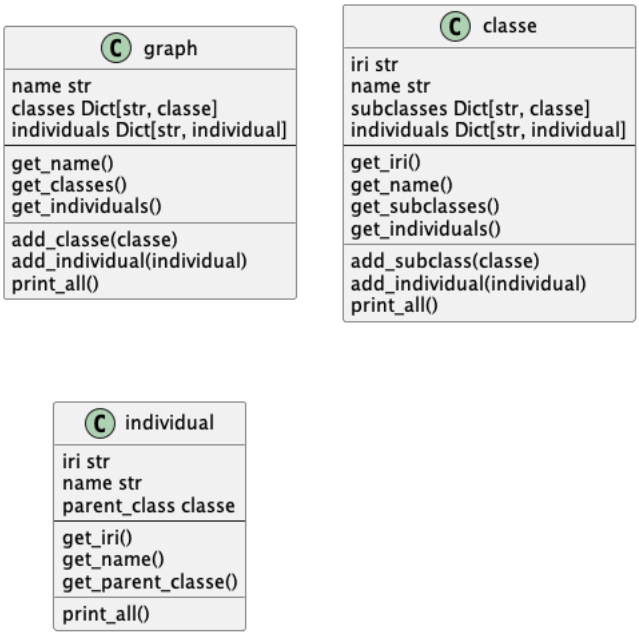

Fig. 1. Graph structure

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

Fig. 2. Example of Subclass triples obtained

The implementation of the program was done using the programming language Python. Furthermore, a part of the manipulation of the OWL file was developed re-utilizing functions of the Python library called OWLREADY2 [5], which also offers the possibility to execute SPARQL queries over the ontology. Regarding the inference, the reasoner Hermit [6] was the chosen one. For visualizing the obtained RDF triples in a graphical representation, the Python package Graphviz [7] was used. However, when the file is too large, the picture is hard to read, and the graph is often stretched. The application Neo4J with the plugin Neosemantics was also utilized to load the output triples and represent them in a graph. The transformation process is represented in the figure 3.

[5] OWLREADY2, documentation, https://owlready2.readthedocs.io/en/v0.37/
[6] Hermit, http://www.hermit-reasoner.com/
[7] Graphviz, https://graphviz.org/

Received April 2023 Accepted July 2023; Published August 2023

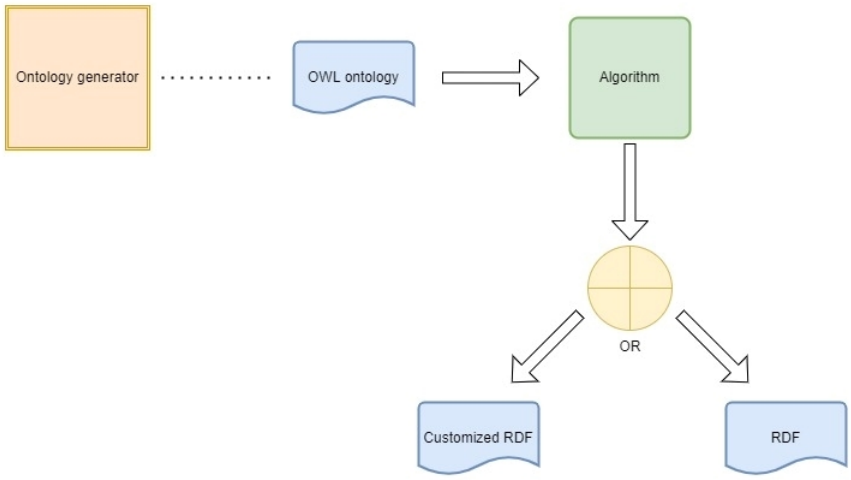

Fig. 3. Diagram of the transformation process

3 Case of study: application on the chess domain

3.1 The ontology

In an attempt of analyzing the existent knowledge formalization done on the chess domain, the work done by Adila Krisnadhi and Pascal Hitzler on their published chapter "Modeling With Ontology Design Patterns: Chess Games As a Worked Example" [6] was found very interesting. However, the design of the aforementioned ontology is more oriented to the representation of a chess competition.

Regarding this case study, the first step was to create an ontology with some individuals (an OWL KG), based on the chess domain. The idea of the design is to represent the game, with all the important factors, and also represent a match that has been played as a list of movements, or in other words, as the evolution of the pieces on the board. As a supplementary support, the rules of the game could be found in the internet [8].

A description of the structure of the created ontology is as follows: The ontology has five main classes which are: "Board", "Match", "Pieces", "Players" and "Rules". The "Board" class contains the sub-classes "Cell", "file" and "rank", with the objective of representing each position where a piece can be placed. The first subclass contains sixty-four individuals in order to instantiate this, and each one of these individuals is related through two object properties to one

[8] https://en.wikipedia.org/wiki/Chess

Received April 2023 Accepted July 2023; Published August 2023

individual of the subclass "file" and one of the subclass "rank". They are also related to a string value, so for example, the cell c3 is related to the string value "c3". This was done to workaround the following issue: the reasoner engine is not capable to understand semantically the name of a subclass or an individual. The "Match" class is designed to represent a specific list of movements that are attached to one specific game played by two entities. An individual of the match is going to be related to an object property with two different players, which one will be identified with the white pieces, meanwhile the other one is going to be related with the black pieces (both sides of the match). The class "Pieces" contains all the different pieces that are part of the game, like the Bishop, the Tower or the King as sub-classes. Each one of this, has individuals to represent the specific pieces that are on the board. For example, the subclass "knight" has two individuals per color. "Players" contains two sub-classes called "AI" and "Person", and they represent the entities that will play the match. "Rules" class contains some specific allowed movements (for example "en_passant", the castle, the promotion) and also the conditions that could get the game to the end in the subclass "win_conditions".

The class hierarchy of the ontology is graphically represented in Protege in the figure 4 and in the corresponding VOWL diagram in figure 5

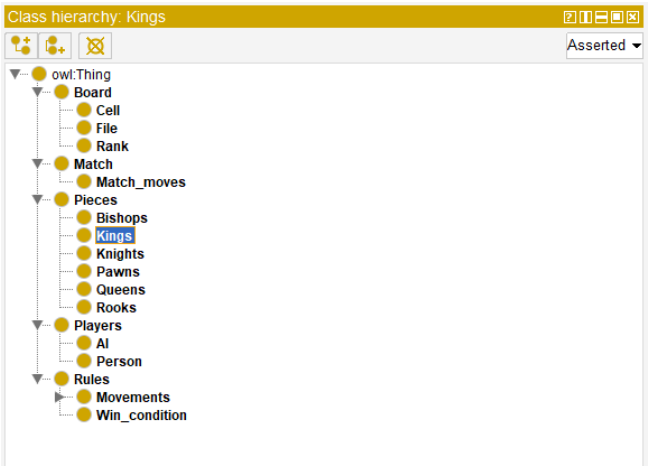

Fig. 4. Protégé structure

The main relations which help to represent a played game are the following:

- Match *Where_black_player_is* Player
- Match *Where_white_player_is* Player
- Match_moves *To_cell* cell
- Match_moves *Moving_Piece* Pieces
- Match_moves *Next_move* Match_moves

Received April 2023 Accepted July 2023; Published August 2023

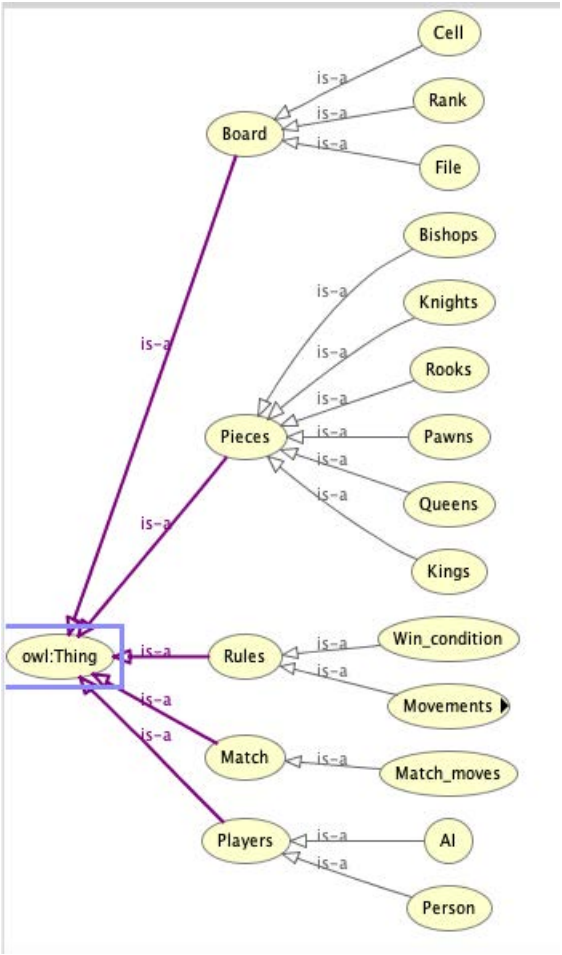

Fig. 5. VOWL diagram of the class hierarchy

– Pieces *Strat_cell_is* Cell

The object property hierarchy of the ontology is described in figure 6.

In order to represent a match, the design takes into account the following statements: A game is played by two players. One plays the white pieces and the other plays the black pieces. A match is a set of movements alternatively of white pieces and black pieces from a starting situation to the end of the match. A movement, in our chess ontology, is basically a piece which is moving from a specific cell to another specific cell of the board.

This ontology was created utilizing the free and open source software Protégé [9], which is widely used to create ontologies and it has a lot of interesting utilities.

[9] Protégé, Homepage https://protege.stanford.edu/ Last accessed 5/6/2022

Received April 2023 Accepted July 2023; Published August 2023

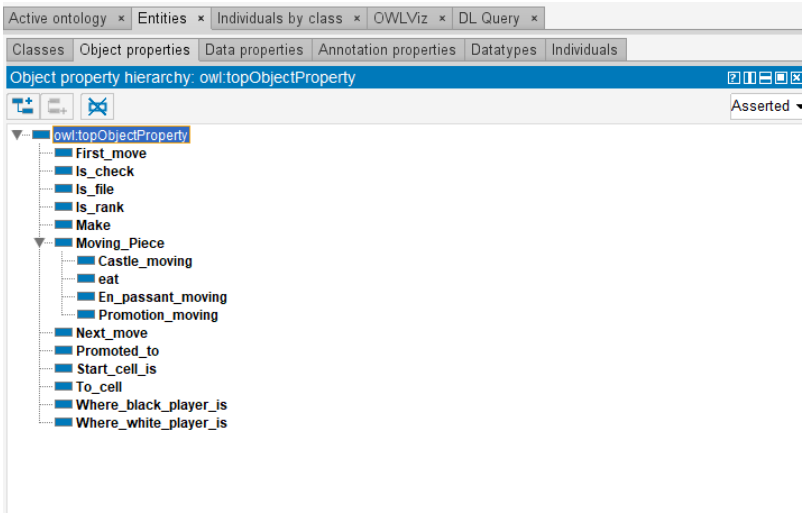

Fig. 6. Object properties in Protégé

The mentioned ontology has been stored in OWL format, since it's one of the most common languages for this purpose, and it has a high level of schema formalization, so it represents the lack of flexibility that it's needed for the objective of this work. This file format can be open as a text file or with different specific software like Protégé. A little part of the structure of the language is shown in figure 7 and in figure 8.

```
    <!-- http://www.semanticweb.org/killi/ontologies/2022/0/Chess_Ontology#Bishops -->

    <owl:Class rdf:about="http://www.semanticweb.org/killi/ontologies/2022/0/Chess_Ontology#Bishops">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/killi/ontologies/2022/0/Chess_Ontology#Pieces"/>
        <owl:disjointWith rdf:resource="http://www.semanticweb.org/killi/ontologies/2022/0/Chess_Ontology#Kings"/>
        <owl:disjointWith rdf:resource="http://www.semanticweb.org/killi/ontologies/2022/0/Chess_Ontology#Knights"/>
        <owl:disjointWith rdf:resource="http://www.semanticweb.org/killi/ontologies/2022/0/Chess_Ontology#Pawns"/>
        <owl:disjointWith rdf:resource="http://www.semanticweb.org/killi/ontologies/2022/0/Chess_Ontology#Queens"/>
        <owl:disjointWith rdf:resource="http://www.semanticweb.org/killi/ontologies/2022/0/Chess_Ontology#Rooks"/>
        <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">This is a piece. There are two bishops of
each side (black and white). See also Bishop&apos;s movement.</rdfs:comment>
    </owl:Class>
```

Fig. 7. Bishop class in OWL file

In order to visualize the ontology in the tool Protégé, the plugins OWLViz [10] and VOWL [11] were utilized. A VOWL representation of the ontology can be seen in the figure 9.

[10] OWLViz, https://protegewiki.stanford.edu/wiki/OWLViz
[11] VOWL, http://vowl.visualdataweb.org/

Received April 2023 Accepted July 2023; Published August 2023

```
    <!-- http://www.semanticweb.org/killi/ontologies/2022/0/chess-ontology-2#Moving_Piece -->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/killi/ontologies/2022/0/chess-ontology-2#Moving_Piece">
        <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/killi/ontologies/2022/0/chess-ontology-2#Match_moves"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/killi/ontologies/2022/0/Chess_Ontology#Pieces"/>
        <rdfs:comment>Hadrien : It is not a functional property because the caste movement &quot;switch and move&quot;
both the King and the Rook</rdfs:comment>
    </owl:ObjectProperty>
```

Fig. 8. Moving_Piece property in OWL file

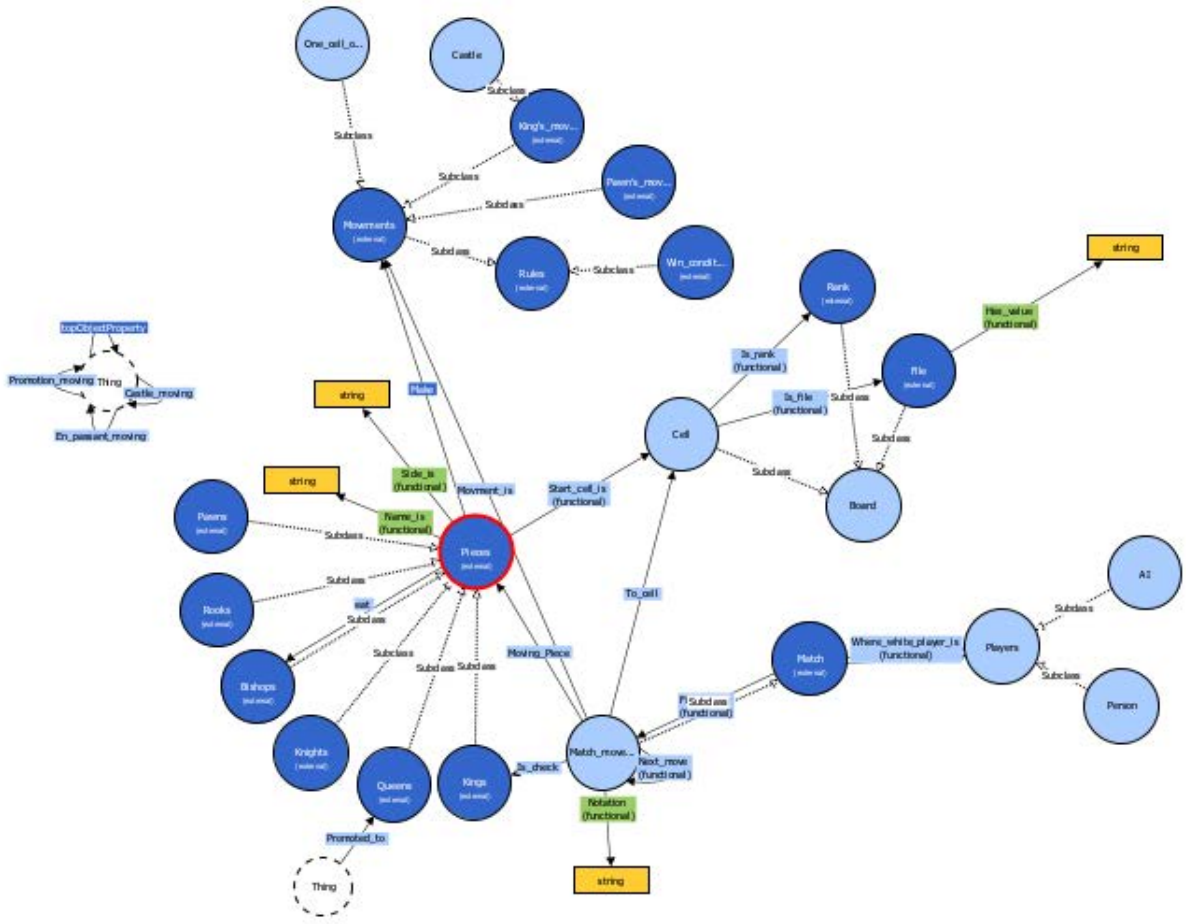

Fig. 9. VOWL ontology representation

3.2 From OWL to RDF

The second step of this work consists in the application of the transformation described in section 2, with the objective of converting the OWL KG file into

Received April 2023 Accepted July 2023; Published August 2023

RDF triples that represent an RDF KG. The created chess KG mentioned in the above section was exported from Protégé as a .OWL file and different tests were executed in order to evaluate the different possible behaviours of the process. The transformation was done with the original ontology file and with the inferenced one, and several outputs were generated formatted as RDF triple stores, customized RDF and RDF/XML.

In order to obtain the knowledge and being able to process it, the OWL structure was transformed into the python graph structure. This load was then evaluated through a comparison between the number of elements that were present in Protégé and the number of elements that were loaded to the python graph structure. In this case, this test had successful results. The figure 10 shows an example where the number of individuals is compared.

Fig. 10. Comparison of individual between Protégé and program

The first output is generated in triples of the form (subject, predicate, object), which represent the relation between two nodes in the knowledge graph. Futhermore, for the sake of the analysis, a customized RDF structure output has been generated, in which the triples are compound. This means that everything from the OWL input is present in the output, concatenating information in the predicate or in the object. As an example, the input OWL statement *pawn promoted to "Pieces not(Kings) not(Pawns)"* was translated to an "RDF" triple with a compound object *"not(Kings) not (Pawns)"*. After cleaning the customized RDF output, the triples are also translated to an RDF structure, which is described in the figure 2. The obtained triples in this occasion have a lack of expressiveness, since there's an obvious schema level difference between it and OWL. This makes that the constraints, types of relationships and all the characteristics that are specific from OWL, are not represented in RDF. Examples of the RDF triples obtained are shown in the figures 11 and 12.

In order to show the lost semantics that this transformation has, the OWL statement *pawn promoted to "Pieces not(Kings) not(Pawns)"* can be highlighted as an example again. In this case, the object or range is only one element for the OWL file, but it cannot be transformed into only one element in the RDF structure. Moreover, it would be necessary to create several triples to represent only

Received April 2023 Accepted July 2023; Published August 2023

```
======================================================================
==================  SUBCLASS TRIPLES ==================
======================================================================
('Chess_Ontology.AI', 'subClassOf', 'Chess_Ontology.Players')
('Chess_Ontology.Castle', 'subClassOf', "Chess_Ontology.King's_movement")
('Chess_Ontology.Cell', 'subClassOf', 'Chess_Ontology.Board')
('Chess_Ontology.Double_forward', 'subClassOf', "Chess_Ontology.Pawn's_movement")
('Chess_Ontology.Eat', 'subClassOf', "Chess_Ontology.Pawn's_movement")
('Chess_Ontology.En_passant', 'subClassOf', "Chess_Ontology.Pawn's_movement")
('Chess_Ontology.Match_moves', 'subClassOf', 'Chess_Ontology.Match')
('Chess_Ontology.One_cell_one_piece', 'subClassOf', 'Chess_Ontology.Movements')
('Chess_Ontology.One_forward', 'subClassOf', "Chess_Ontology.Pawn's_movement")
('Chess_Ontology.One_square_move', 'subClassOf', "Chess_Ontology.King's_movement")
('Chess_Ontology.Person', 'subClassOf', 'Chess_Ontology.Players')
('Chess_Ontology.Promotion', 'subClassOf', "Chess_Ontology.Pawn's_movement")
("Chess_Ontology.Queen's_movement", 'subClassOf', 'Chess_Ontology.Movements')
('Chess_Ontology.Bishops', 'subClassOf', 'Chess_Ontology.Pieces')
('Chess_Ontology.Kings', 'subClassOf', 'Chess_Ontology.Pieces')
('Chess_Ontology.Knights', 'subClassOf', 'Chess_Ontology.Pieces')
('Chess_Ontology.Movements', 'subClassOf', 'Chess_Ontology.Rules')
('Chess_Ontology.Pawns', 'subClassOf', 'Chess_Ontology.Pieces')
('Chess_Ontology.Queens', 'subClassOf', 'Chess_Ontology.Pieces')
('Chess_Ontology.Rooks', 'subClassOf', 'Chess_Ontology.Pieces')
('Chess_Ontology.Win_condition', 'subClassOf', 'Chess_Ontology.Rules')
("Chess_Ontology.Bishop's_movement", 'subClassOf', 'Chess_Ontology.Movements')
("Chess_Ontology.King's_movement", 'subClassOf', 'Chess_Ontology.Movements')
("Chess_Ontology.Knight's_movement", 'subClassOf', 'Chess_Ontology.Movements')
("Chess_Ontology.Pawn's_movement", 'subClassOf', 'Chess_Ontology.Movements')
("Chess_Ontology.Rook's_movement", 'subClassOf', 'Chess_Ontology.Movements')
```

Fig. 11. Example of Subclass triples obtained

one OWL relation with constraints. An example of a graphical representation of the generated triples is in the figure 13.

4 Learning lessons and further steps

Through this exercise, the authors have analyzed the lost of expressiveness that takes place in a simple transformation from an OWL KG into an RDF KG, which is related to the difference over the schema level of knowledge representation between them. RDF is a lighter and more flexible representation, and it's not designed to express the level of semantics that takes into account knowledge like type of relationships (symmetric, transitive, etc.), neither constraints (some, all, etc.), between others.

Currently, the authors have in mind two possible options to treat this schema level difference. The first one is to create a representation of the OWL ontology without constraints, but maintaining all the logic that is behind these rules or definitions. A possible path to do this is investigating and testing the output generated after the inferences over the OWL structure and deleting the con-

Received April 2023 Accepted July 2023; Published August 2023

```
================================================================
================== INDIVIDUALS TRIPLES ==================
================================================================
('Chess_Ontology.B1', 'individualOf', 'Chess_Ontology.Bishops')
('Chess_Ontology.c1', 'individualOf', 'Chess_Ontology.Cell')
('Chess_Ontology.B2', 'individualOf', 'Chess_Ontology.Bishops')
('Chess_Ontology.f1', 'individualOf', 'Chess_Ontology.Cell')
('Chess_Ontology.B3', 'individualOf', 'Chess_Ontology.Bishops')
('Chess_Ontology.c8', 'individualOf', 'Chess_Ontology.Cell')
('Chess_Ontology.B4', 'individualOf', 'Chess_Ontology.Bishops')
('Chess_Ontology.f8', 'individualOf', 'Chess_Ontology.Cell')
('Chess_Ontology.K1', 'individualOf', 'Chess_Ontology.Kings')
('Chess_Ontology.e1', 'individualOf', 'Chess_Ontology.Cell')
('Chess_Ontology.K2', 'individualOf', 'Chess_Ontology.Kings')
('Chess_Ontology.e8', 'individualOf', 'Chess_Ontology.Cell')
('Chess_Ontology.N1', 'individualOf', 'Chess_Ontology.Knights')
('Chess_Ontology.b1', 'individualOf', 'Chess_Ontology.Cell')
('Chess_Ontology.N2', 'individualOf', 'Chess_Ontology.Knights')
```

Fig. 12. Example of Individual triples obtained

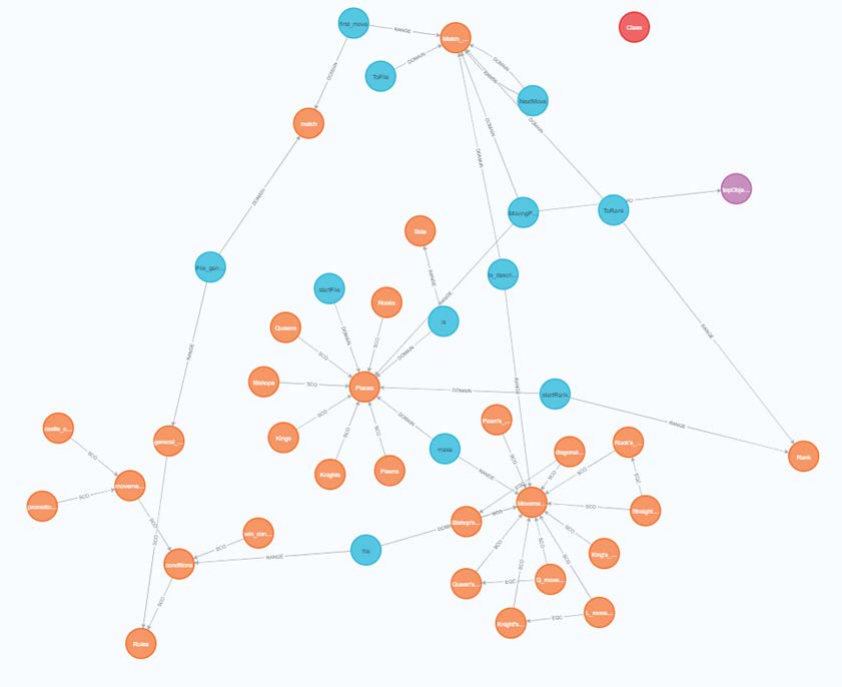

Fig. 13. Output triples in Neo4j

straints. The second option is to keep all the constraints and finding a way to express them. This could represent the creation of new nodes, and additional

Received April 2023 Accepted July 2023; Published August 2023

triples would represent the complex axioms in RDF. With the customized RDF output, a primitive and first attempt of maintaining all the characteristics in the RDF KG was done, but this is only with analysis purposes since it's not practical as its. However, this output will be also taken into account in the further steps.

Regarding the next steps, one possible path would be to add a new step in the program with the objective of interpreting this customized RDF structure, giving to the compound triples a representation that fits with the RDF syntax. Another possible path to follow, in order to add these characteristics to an RDF graph, could be to add a layer of representation, to express this structure in an efficient manner. The authors will research existent work related to adding formalization to RDF KG. It's interesting for the authors to study Shapes Constraint Language [1], the characteristics of the property graphs [9], ontology alignment [2] and the state of the art of its applications into the knowledge graphs. The objective would be to integrate some of this existent concepts, or develop a new one inspired by them, and introduce it in this transformation process. Furthermore, it's interesting for the authors to investigate the transformation in the opposite direction, which means from an RDF KG to an OWL KG.

References

1. Shapes constraint language (SHACL). Tech. rep., W3C (Jul 2017), https://www.w3.org/TR/shacl/
2. Ardjani, F., Bouchiha, D., Malki, M.: Ontology-alignment techniques: Survey and analysis. International Journal of Modern Education and Computer Science **7**, 67–78 (11 2015). https://doi.org/10.5815/ijmecs.2015.11.08
3. Baget, J.F., Chein, M., Croitoru, M., Fortin, J., Genest, D., Gutierrez, A., Leclère, M., Mugnier, M.L., Salvat, E.: RDF to Conceptual Graphs Translations. In: CS-TIW: Conceptual Structures Tool Interoperability Workshop. vol. LNAI, p. 17. Springer, Moscow, Russia (Jul 2009), https://hal-lirmm.ccsd.cnrs.fr/lirmm-00410621
4. Desouki, A., Röder, M., Ngonga Ngomo, A.C.: Ranking on very large knowledge graphs. pp. 163–171 (09 2019). https://doi.org/10.1145/3342220.3343660
5. Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., Melo, G.d., Gutierrez, C., Kirrane, S., Gayo, J.E.L., Navigli, R., Neumaier, S., Ngomo, A.C.N., Polleres, A., Rashid, S.M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., Zimmermann, A.: Knowledge Graphs. Synthesis Lectures on Data, Semantics, and Knowledge **12**(2), 1–257 (Nov 2021). https://doi.org/10.2200/S01125ED1V01Y202109DSK022, https://www.morganclaypool.com/doi/abs/10.2200/S01125ED1V01Y202109DSK022, publisher: Morgan & Claypool Publishers
6. Krisnadhi, A., Hitzler, P.: Modeling With Ontology Design Patterns: Chess Games As a Worked Example, vol. 25, chap. 1, p. 3–21. IOS Press (2016). https://doi.org/10.3233/978-1-61499-676-7-3
7. Lera, I., Juiz, C., Puigjaner, R.: Performance-related ontologies and semantic web applications for on-line performance assessment of intelligent systems. Science of Computer Programming **61**(1), 27–37 (2006). https://doi.org/https://doi.org/10.1016/j.scico.2005.11.003, https://www.sciencedirect.com/science/article/pii/S016764230600013X, special Issue on Quality system and software architectures

Received April 2023 Accepted July 2023; Published August 2023

8. Noy, N., Gao, Y., Jain, A., Narayanan, A., Patterson, A., Taylor, J.: Industry-scale knowledge graphs: lessons and challenges. Communications of the ACM **62**(8), 36–43 (Jul 2019). https://doi.org/10.1145/3331166, https://doi.org/10.1145/3331166
9. Tomaszuk, D., Angles, R., Thakkar, H.: Pgo: Describing property graphs in rdf. IEEE Access **8**, 118355–118369 (2020)
10. Wood, D., Lanthaler, M., Cyganiak, R.: Rdf 1.1 concepts and abstract syntax. W3C Recommendation, W3C (2014)

Received April 2023 Accepted July 2023; Published August 2023