IA301 : Logics and Symbolic AI

Ontology Protege Project Report

Football Mercato Ontology : an ontology to help professional football recruiters

Maxime Lhoustau – Oumar Sekou Camara

Football Mercato Ontology Project

Introduction :

We wanted to build an ontology about the football world. We have been inspired by all the datasets we could find on the Internet about players, and we thought that we could use it to build some type-profiles of players. These type-profiles have been built subjectively based on our understanding of the football game, and can be extended to more knowledge if someone has the competency to do it. Our project is in fact to resolve a classification problem.
We used Protege software to design our ontology. We used features of Protege that need plugins to work (Cellfie for the Excel importation, OWLViz to visualize data, SWRL rules to achieve data comparisons…). That is why this ontology requires the Pellet reasoner to infer our individuals.

Maxime Lhoustau – Oumar Sekou Camara

MIRO : Minimum Information for Reporting an Ontology

A) Basics

1. Ontology Name

Football Mercato Ontology (FMO), v1.0

2. Ontology Owner

Maxime Lhoustau and Oumar Sekou Camara

3. Ontology License

Free of using, sharing, improving the ontology with mention of the owners.

4. Ontology repository

The ontology can be found on the following github repository :
https://github.com/maximelhoustau/Symbolic_AI
It contains :

- The ontology FMA
- 2 Excels sheets containing the clubs database and the players database
- 1 json file containing Cellfie rules to add all the clubs and players properties from the Excel files to the ontology

B) Motivation

1. Need

The goal of this ontology is to help professional football players recruiters based on a database of players and clubs. The situation can be viewed as followed :
It exists a lot of statistics of professional football players in databases that can be found on internet. We can cite for example datasets from the FIFA video games or from the Opta Sports company. These statistics can be used to classify all the players onto different type-profiles useful for the recruiters. Indeed, these type-

profiles could be build by themselves in order to match with their research. The ontology would be a helpful tool to classify and better targeting players during the mercato in the in-between seasons.

2. Competition

A few ontologies about football players exist, such as this one :
http://fr.dbpedia.org/ontology/SoccerPlayer
But basically this ontology doesn't rely on players statistics and give a summary about their general informations. No type-profiles are inferred in contrary of our ontology, which is the key point of our project.

3. Target audience

This ontology should be used by professional football recruiters to help them refine their research. But it can also be used by everyone working or interested into the professional football world and the classification of players based on their statistics.

C) Scope, requirements, development community

1. Scope and coverage

The field of interests of this ontology is professional football, including leagues, clubs, players and their statistics. Here, the term statistics means that we gave some data properties to the objects of the ontology that refer to real statistics data that are not in our project. This can of course be extended to more data.

The ontology is used here to infer type-profile players based on different informations contained in the Players class. We gathered objectives data to infer subjectives profiles such as the following examples :

- If a player is playing at the right back position and at the left back position then he is an adaptable lateral
- If a player is playing center back and he is more than 28 years old and he is playing in a top club, then he is an experienced center back

The granularity of representation is the individuals that constitute the PlayersToClassify set of players.

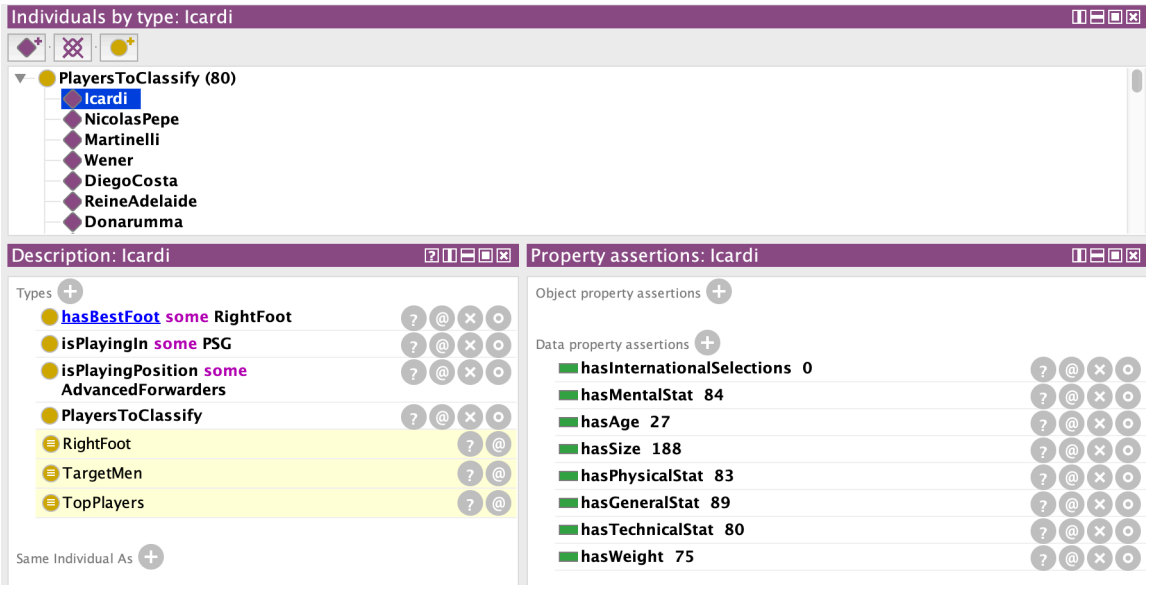

Figure 1: Screenshot of Protege - Individual players and their properties

2. Development community

The group of people that actually create the content of the ontology are the institutions responsible for professional football (FIFA) and the institutions that gather the datas (profesionnal, open source, or private institutions).

D)Knowledge acquisition

1. Knowledge acquisition methodology

We used a feature of the Protege software to automatically fill our ontology with players, clubs and their properties (data or class properties) from Excel sheets. We wrote a Json script that enable to specify which column corresponds to which property. The Json file is contained in the repository.

To use this feature, you should go the Tools > Create Axioms from Excel workbook and then use the Protege pan to choose an Excel sheet and a Json file to run.

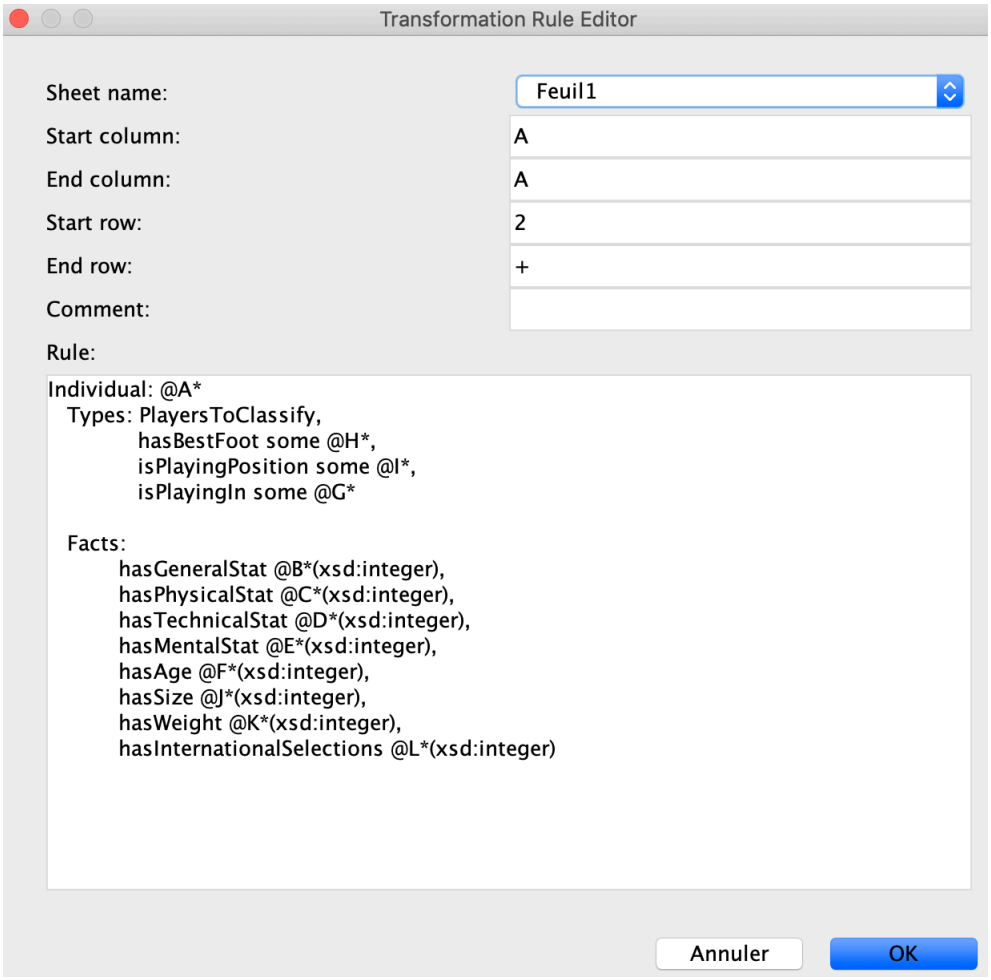

Figure 2: Json script to automatically add Individuals to the ontology

2. Source knowledge location

We used this dataset as an example to create the methodology of the players properties : https://www.kaggle.com/stefanoleone992/fifa-20-complete-player-dataset
This dataset contains content from the FIFA20 video games which is a football simulation based on the real teams and players. The game contains a lot of datas every year and is now considered as a standard source of real life datas.

E) Ontology Content

1. Knowledge representation language

OWL version 2

2. Development requirement

Protege version 5.5

3. Ontology metrics

Here are the ontology metrics of the FMO :

Object property axioms

SubObjectPropertyOf	7
EquivalentObjectProperties	0
InverseObjectProperties	3
DisjointObjectProperties	0
FunctionalObjectProperty	3
InverseFunctionalObjectProperty	3
TransitiveObjectProperty	0
SymmetricObjectProperty	0
AsymmetricObjectProperty	0
ReflexiveObjectProperty	0
IrrefexiveObjectProperty	0
ObjectPropertyDomain	8
ObjectPropertyRange	9
SubPropertyChainOf	0

Data property axioms

SubDataPropertyOf	9
EquivalentDataProperties	0
DisjointDataProperties	0
FunctionalDataProperty	9
DataPropertyDomain	12
DataPropertyRange	10

Metrics

Axiom	1551
Logical axiom count	1362
Declaration axioms count	180
Class count	76
Object property count	11
Data property count	12
Individual count	80
Annotation Property count	4

Class axioms

SubClassOf	252
EquivalentClasses	37
DisjointClasses	2
GCI count	89
Hidden GCI Count	37

Individual axioms

ClassAssertion	355
ObjectPropertyAssertion	0
DataPropertyAssertion	640
NegativeObjectPropertyAssertion	0
NegativeDataPropertyAssertion	0
SameIndividual	0
DifferentIndividuals	0

Annotation axioms

AnnotationAssertion	9
AnnotationPropertyDomain	0
AnnotationPropertyRangeOf	0

4. Entity naming convention

- Classes :
  We used a capital letter for each letter of a begining word, as well as the plurial and no space between words for every classes of classification.
- Individuals :
  We used the name and surname of players as an identifier for each of them. We used no spaces between name and surname.

- Data and Object properties :
  We used the CamelCase naming convention, and ever properties start with hasXXX or isXXX based on the functionality of the properties.

5. Identifier generation policy

As said in the above section, we used the name and surname of players as an unique identifier to the individuals. We could have used a (mm :hashEncode) tag in the Json file to provide a unique identifier to the individuals based on the Excel cell values (see https://github.com/protegeproject/cellfie-plugin/wiki/Grocery-Tutorial Step 6 for more informations).

6. Entity metadata policy

Each class minimally requires a label and a text description (see the ontology for more details).

7. Ontology relationships

As said before, we built our own profiles (corresponding to classes in the ontology) according to our understanding and our comprehension of the football world. The rules between classes and subclasses are therefore subjectives. The objective here was not to build objectives profiles but to show that it is possible to bring subjective knowledge and apply it to the football world.

We defined several families of classes :

- Clubs : these classes contains every football clubs from different european national championships used in the ontology. They are used to record in which club the players are actually playing. We ranked the clubs based on a data property called hasPrestige that represents the prestige of the club. As class expressions doesn't support data comparisons (especially the HermiT reasoner), we used 3 subclasses to rank the clubs according their prestige : TopClubs (hasPrestige >=4), RegularClubs (hasPrestige = 3) and BadClubs (hasPrestige <=2).
- Leagues : these classes contains the national championships in which the clubs are playing. It is not really useful here but it could be extended to more knowledge, based on a ranking of the championships. The queries can also include leagues but we did not use it into the properties between classes.
- Players : these classes contains every players to classify into the ontology. The players are classified whether according the position they are playing on the field, whether according the profile inferred by the reasoner.

- PlayersCaracteristics : groups the Positions, Profiles and BestFoot for an easier display of these classes. They are the same classes that are contained into the Players classes since they are subclasses of Players and PlayersCaracteristics.

We used object properties to link objects between them :

- isPlayingInLeague : to specify in which national league a club is playing into (hasClub being the opposite property)
- isPlayingPosition : to specify to role of players on the field.
- hasBestFoot : to specify whether the player is right or left legged (isTheBestFootOf being the opposite property).
- isPlayingIn : to specify in whixh club a player is actually playing (isTheClubOf being the opposite property).

We also used data properties to link statistics to their entity into the ontology :

- hasPrestige : prestige of a club on the international scene (integer between 1 and 5)
- hasAge : age of a player (integer)
- hasInternationalSelections : state if the player is playing in a national team or not, and if yes contains the number of selections (integer between 0 and 300)
- hasSize : size of a player (integer between 150 and 250 centimeters)
- hasWeight : weight of a player (integer between 60 and 120)
- hasGeneralStat : general stat of a player (integer between 0 and 100)
- hasTechnicalStat : technical stat of the player (integer between 0 and 100)
- hasMentalStat : mental stat of the player (integer between 0 and 100)
- hasPhysicalStat : physical stat of the player (integer between 0 and 100)

For more informations about relations between classes (and profiles), I joined a text file (profiles.txt) containing the relations in natural language (French).

8. Axioms patterns

We used a special pattern to design our ontology. In fact, we designed classes such as, deeper we are in the subclasses, and more precise are the entities belonging to it.
For example, a player whose role is LeftBacks belong to the LateralPlayers classes, which belong to the Defenders grouping every defending position of football.

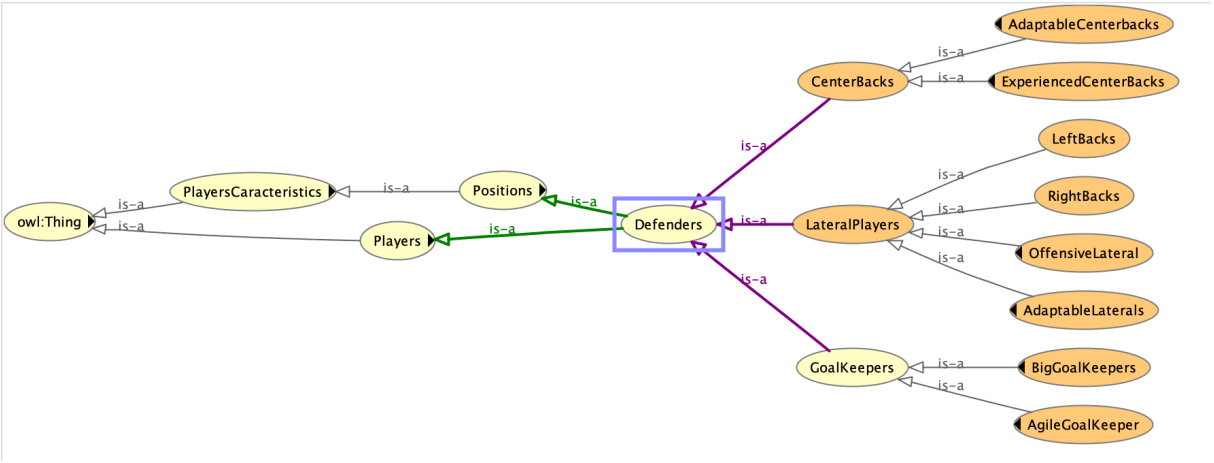

Figure 3 : Screenshot of the OWLViz tab, example of pattern into the Defenders players

Every classes we build have this pattern. To visualize more of our pattern, please refer to the OWLViz tab to see every groups of classes.

F) Quality assurance

1. Testing

To test our ontology, we built a special column on our Excel sheet containing datas of players. The special column contains the profiles that the player should be inferred to into the ontology. When we extracted data from Excel to Protege, we made the reasoner work and we looked at the classes inferred. If it was the right class then we marked the cell in green, otherwise we marked it in red.
We had been able to successfully modify the ontology to infer every player to their right profile classes.

2. Evaluation

We have been able to infer our players into every type-profiles we built. Since these are subjectives, we can say that there isn't an obvious way to evaluate our ontology. However, we think that we achieve our goal because the testing went well and that the individuals seem to be inferred in the right classes.

Figure 4 : Indiciduals inferred into profile classes

3. Example of use

One can simply use our project by adding its own database to the ontology and see the profiles inferred (like Figure 4).
We can also use the DL Query tab of Protege to build special queries. For example, here is an example to display LeftBacks players who are less than 25 years-old.

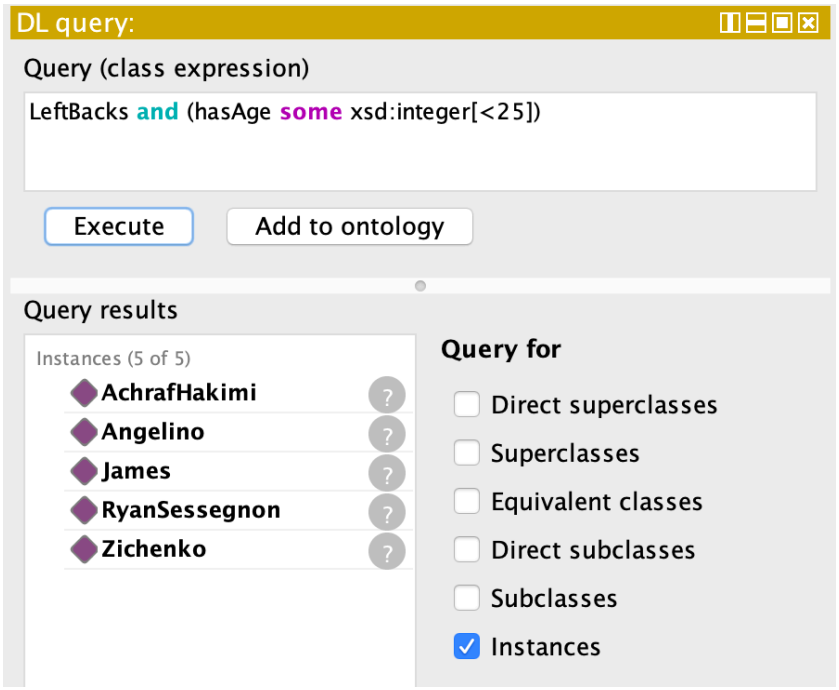

Figure 5: Screenshot of a DL query

Appendice – SWRL rules :

We wanted to build classes according to a comparison between stats of one player :

- If the maximum statistics of one player is the physical one, then he is a PhysicalPlayers
- If the maximum statistics of one player is the technical one, then he is a TechnicalPlayers
- If the maximum statistics of one player is the mentalone, then he is a LeaderPlayers

These classes would be really interesteing since we could look for a player playing in a special position, plus having special skills like technical or mental ones.

To achieve comparisons, we had to deal with SWRL rules. As we can see for the LeaderPlayers, there isn't any rule to infer that class :

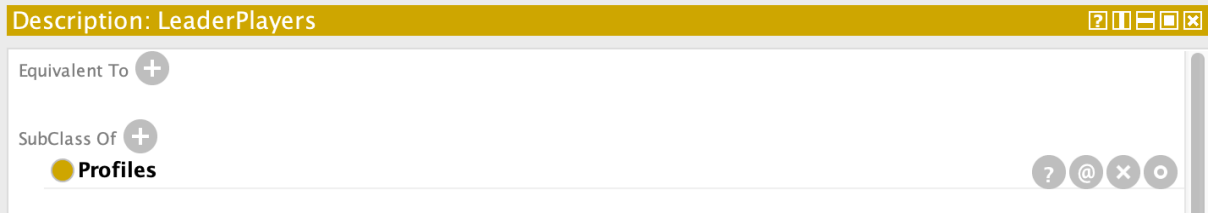

Figure 6 : LeaderPlayers class is empty

We had to build special SWRL rules with built-ins, like for example this rule for TechnicianPlayers :

```
Rule: Players(?player), hasTechnicalStat(?player, ?t_stat), hasPhysicalStat(?player,
?p_stat), hasMentalStat(?player, ?m_stat), greaterThan(?t_stat, ?p_stat),
greaterThan(?t_stat, ?m_stat) -> TechnicalPlayers(?player)
```

Each class have its own rule that we can find in the view Window > Views > Ontology views > Rules.

This is why our ontology doesn't work with the HermiT reasoner, but with Pellet reasoner (because HermiT doesn't support SWRL rules). It is still possible to use the HermiT reasoner but you have to delete these rules first (and the inferred classes won't work anymore).