University of Liege

2023-2024

INFO9014-1 - Knowledge Representation and Reasoning project

Milestone 2

Wilfried Mvomo Eto - s226625

May 15, 2024

Context

To transition from a physically centralized design to a semantically centralized one, facilitating the diverse needs of our relational database across departments, ensuring interoperability of semantics is essential. This pivotal phase primarily focuses on crafting an ontology to streamline the retrieval of information within a timely manner. Our aim is to simplify query formulation, eliminating the complexity users previously encountered with the relational database designed in milestone 1.

1 Description of the Universe of Discourse

We introduce the ontology design using Description Logics, which enables us to describe the universe of discourse in terms of concepts, roles, and individuals. The most relevant concepts identified are: **PractitionerIdentity**, **FairPlayIndex**, **Team**, **OriginCountry**, **Location**, **SportsFacility**, and **Penalties**, which are disjoint. Some of them have a subclasses :

- Subclasses of **PractitionerIdentity**: `PlayerIdentity`, `RefereeIdentity` and `ManagerIdentity`;
- Subclasses of **FairPlayIndex**: `FairPlayPlayer` and `FairPlayTeam`;
- Subclasses of **Penalties**: `PlayerPenalties` and `TeamPenalties`;

`FairPlayIndex` represents a category or class related to fair play index within the universe of discourse, `QualificationZone` represents a concept within the ontology that pertains to zones or regions related to qualification in a specific context;`Location` represents a concept within the ontology the place of birth of practitioner and `OriginCountry` represents a concept within the ontology that the native country of practitioner.

The roles are defined as follows:

- $ManagerIdentity$ "coaches" some $Player$;
- $ManagerIdentity$ "managerAttendsIn" some $QualificationZone$;
- $RefereeIdentity$ "refereeAttendsIn" some $QualificationZone$;
- $PlayerIdentity$ "playerAttendsIn" some $QualificationZone$;
- $SportsFacility$ "isLocatedIn" $Location$;
- $FairPlayIndexPlayer$ "gottenBy" some $Player$;
- $FairPlayIndexTeam$ "obtainedBy" some $Team$;
- $PlayerPenalties$ "achievedByPlayer" some $Player$;
- $TeamPenalties$ "achievedByTeam" some $Team$;
- $ManagerIdentity$ "manages" some $Team$.
- $PlayerIdentity$ "playsFor" some $Team$;
- $ManagerIdentity$ "managerIsBornIn" some $Location$;
- $RefereeIdentity$ "refereeIsBornIn" some $Location$;
- $PlayerIdentity$ "playerIsBornIn" some $Location$;
- $ManagerIdentity$ "managerComesFrom" some $Location$;

- $RefereeIdentity$ "refereeComesFrom" some $Location$;
- $PlayerIdentity$ "playerComesFrom" some $Location$;
- $SportsFacility$ "isLocatedIin" some $Location$;
- $SportsFacility$ "belongsToTeam" some $Team$;
- $Team$ "teamIsLocated" some $Location$;
- $Location$ "townOf" some $OriginCountry$;
- $PlayerIdentity$ "playsAtHome" some $Stadium$;

Finally, individuals, for instance, can be $PlayerIdentity(Marcello)$; $ManagerIdentity(Alpha)$; $Location(Ourdjar)$; $OriginCountry(Canada)$; $coaches(Alpha, Marcello)$.

2 Knowledge base engineering

2.1 ALC concept descriptions

On the previous concepts and roles above mentioned which represent respectively classes and properties in OWL, we rely on the ALC for defining the ALC concept descriptions. They are a very important step to ensure that our ontology will be consistent allowing us to realize advanced reasoning. Thereby, we have several concept descriptions among which: PractitionerIdentity, Team, FairPlayIndex, OriginCountry and Location with their subclasses ; ∃ coaches.PlayerIdentity; ∃ manages.Team; ∃ managerAttendsIn.QualificationZone; ∃ isLocatedIn.Location; ∃ gottenBy.FairPlayIndexPlayer. We can form some other ALC concept descriptions by using a complement or intersection, but the goal is to build those which will help to capture the domain of football that we define in the previous milestone.

2.2 Terminological Box (TBox) and Assertional Box (ABox)

In the domain of ontology engineering, establishing a comprehensive knowledge base $K =< T, A >$ is essential for capturing the intricacies of a given domain. This knowledge base often comprises a set of statements and relationships that define the fundamental concepts and properties within the domain. In this context, we present a curated set of statements that form the backbone of our ontology, encapsulating various aspects of the domain of interest. Each statement represents a logical assertion about the relationships between entities, providing valuable insights into the structure and dynamics of the domain.

Our knowledge base comprises a generalized terminology T, which is a finite set of the following statements:

1. Every manager identity coaches a player identity:
   `ManagerIdentity` $\subseteq \exists$`coaches.PlayerIdentity`.
2. Every Location is town of a origin country: `Location` $\subseteq \exists$ `townOf.OriginCountry`.
3. Every manager identity manages some team : `ManagerIdentity` $\subseteq \exists$`manages.Team`.
4. Every referee identity comes from in origin country:
   `RefereeIdentity` $\subseteq \exists$`refereeComesfrom.OriginCountry`.

5. Every manager identity comes from in origin country:
   $\texttt{ManagerIdentity} \subseteq \exists \texttt{managerComesfrom.OriginCountry}$.
6. Every player identity comes from in origin country:
   $\texttt{PlayerIdentity} \subseteq \exists \texttt{playerComesfrom.OriginCountry}$.
7. Every manager identity attends in qualification zone:
   $\texttt{ManagerIdentity} \subseteq \exists \texttt{managerAttendsIn.QualificationZone}$.
8. Every player identity plays for some team:
   $\texttt{PlayerIdentity} \subseteq \exists \texttt{playsFor.Team}$.
9. Every player identity attends in some qualification zone:
   $\texttt{PlayerIdentity} \subseteq \exists \texttt{playerAttendsIn.QualificationZone}$
10. Every referee identity attends in qualification zone:
    $\texttt{RefereeIdentity} \subseteq \exists \texttt{refereeAttendsIn.QualificationZone}$.
11. Every sport facility is located in a location:
    $\texttt{SportsFacility} \subseteq \exists \texttt{isLocatedIn.Location}$.
12. Every sport facility belongs to a team:
    $\texttt{SportsFacility} \subseteq \exists \texttt{belongsToTeam.Team}$.
13. Every penalty player is achieved by a player identity:
    $\texttt{PlayerPenalties} \subseteq \exists \texttt{achievedByPlayer.PlayerIdentity}$.
14. Every penalty team is achieved by a team identity:
    $\texttt{TeamPenalties} \subseteq \exists \texttt{achievedByTeam.TeamIdentity}$.
15. Every team is located in a location:
    $\texttt{Team} \subseteq \exists \texttt{teamIsLocated.Location}$.
16. Every player's fair play index is gotten by a player identity:
    $\texttt{FairPlayIndexPlayer} \subseteq \exists \texttt{gottenBy.PlayerIdentity}$.
17. Every team's fair play index is obtained by a team:
    $\texttt{FairPlayIndexTeam} \subseteq \exists \texttt{obtainedBy.Team}$.

All the concepts can be instantiated, thereby the concept satisfiability is done. Besides knowledge base satisfiability is done as well.We verified them by relying on the semantic tableau algorithm. .

2.3 Ontology Development: Harnessing OWL 2 RL for Scalable Reasoning and Empowering Knowledge Construction

We have selected certain predicates that require first-order logic to be integrated in order to deduce implicit knowledge within a polynomial time frame. To elucidate the importance of **OWL 2 RL** for our ontology and its coherence with the presented knowledge base, we emphasize the crucial role of a robust querying mechanism capable of efficiently managing large datasets.

By employing OWL 2 RL, we augment the specificity and complexity of constraints that govern the properties and relationships among entities within our ontology. Below are exemplar axioms expressible in OWL 2 RL pertinent to our domain:

1. Existential It allows the use of an existential quantification for specifying that instances of one class must have at least one relationship with instances of another class.

2. Constraint on Entity' names: Leveraging OWL 2 RL, we can specify that each Entity must have a unique identifier. This is achieved by employing the hasKey construct, which defines a key for the Identity class comprising the code properties. By doing so, we ensure that no two Identities share the same combination of data type properties identifying identity. Thereby enhancing data integrity and facilitating more precise identification of entities.

3. It enables the use of first-order logic for inferring new knowledge relying on some predicates.

By using OWL 2 RL, we can formally express these constraints and integrate them into our ontology to ensure its consistency and integrity. These constraints help us define clear rules about the structure and relationships of entities in our domain, making it easier to manage and analyze data in our ontology. The base IRIs for the ontology are:

- **TBox**: `http://www.example.org/foot-infos/ont`
  The TBox contains the classes and properties of the ontology.

- **ABox**: `http://www.example.org/foot-infos/resources`
  The ABox contains the instances of the ontology (e.g. cities, players, teams...)

In the process of converting our data to RDF format, we establish several namespace prefixes for clarity and consistency throughout our ontology. These prefixes serve as shorthand references to specific namespaces, simplifying the representation of URIs. Here are the prefixes we'll employ:

- "`foot`:" stands for the ontology namespace, represented by `http://www.example.org/foot-infos/ont`.

- "`res`:" represents the resource namespace, denoted by `http://www.example.org/foot-infos/resources`.

Main Classes: To facilitate the conversion of our Entity-Relationship Diagram (ERD) into RDF format, we establish key structuring classes.The class forms are in PascalCase and delineate the primary entities outlined in the ERD Figure 2 , namely:

- `foot:PractitionerIdentity`
- `foot:Penalties`
- `foot:FairPlayIndex`
- `foot:QualificationZone`
- `foot: Location`
- `foot:Team`
- `foot: SportsFacility`

These classes symbolize different concepts, hence they are made disjoint. We have the subclasses of some classes mentioned above as well:

For **PractitionerIdentity** class:

- `foot:PlayerIdentity`
- `foot:RefereeIdentity`
- `foot:ManagerIdentity`

For **FairPlayIndex** class:

- `foot:FairPlayIndexPlayer`
- `foot:FairPlayIndexTeam`

For **Penalties** class:

- foot:PlayerPenalties
- foot:TeamPenalties

Notice that FairPlayIndexPlayer and FairPlayIndexTeam are disjoint classes.

Properties: The properties associated with our ontology are named using CamelCase convention:

Relation	Type
foot:coaches	irreflexive
foot:managerAttendsIn	irreflexive, asymmetric
foot:playerAttendsIn	irreflexive, asymmetric
foot:refereeAttendsIn	irreflexive, asymmetric
foot:playerComesFrom	irreflexive
foot:managerComesFrom	irreflexive
foot:refereeComesFrom	irreflexive
foot:belongsToTeam	irreflexive
foot:achievedByTeam	irreflexive, asymmetric
foot:achievedByPlayer	irreflexive, asymmetric
foot:gottenBy	irreflexive, asymmetric
foot:obtainedBy	irreflexive, asymmetric
foot:isIncludedIn	irreflexive
foot:isLocated	irreflexive
foot:manages	irreflexive
foot:playsFor	irreflexive
foot:townOf	irreflexive
foot:teamIsLocated	irreflexive
foot:cityContinent	irreflexive, asymmetric

There are some inverse object properties like countryConsistsOf (inverse of townOf), playerNativeCountry (inverse of playerIsBornIn), managerNativeCountry (inverse of managerIsBorn), refereeNativeCountry (inverse of refereeIsBornIn),holds, managerEmployer ((inverse of playsFor), playerEmployer (inverse of manages), isCoachedby (inverse of coaches), isBorn (inverse of comesFrom), isMadeUp and isIncludedIn.

Data properties: The properties associated with our ontology are also named using CamelCase convention:

Relation	Type
foot:playerFirstName	functional
foot:playerLastName	functional
foot:playerDateOfBirth	functional
foot:refereeFirstName	functional
foot:refereeLastName	functional
foot:refereeDateOfBirth	functional
foot:managerFirstName	functional
foot:managerLastName	functional
foot:managerDateOfBirth	functional
foot:playerNativeCountry	functional
foot:refereeNativeCountry	functional
foot:managerNativeCountry	functional
foot:playerGenre	-
foot:managerGenre	-
foot:refereeGenre	-
foot:teamName	functional
foot:assists	-
foot:goals	-
foot:goalsFor	-
foot:goalsAgainst	-
foot:cityName	functional
foot:countryName	functional
foot:playerYellowCardNumber	-
foot:playerRedCardNumber	-
foot:teamYellowCardNumber	-
foot:teamRedCardNumber	-
foot:position	-
foot:playerFairPlayDate	functional
foot:teamFairPlayDate	-
foot:playerPenaltyDate	functional
foot:teamPenaltyDate	functional
foot:qualificationZoneName	functional
foot:teamQualificationZoneName	functional
foot:teamCountry	asymmetric, irreflexive

Using cityName, countryName, stadiumName“, qualificationZoneName, and teamName as key for Location, OriginCountry, SprotsFacility, QualificationZone and Team. This is for uniquely identifying their respective classes, ensuring clear and unambiguous identification within the ontology.

Overall, the selection of ‘cityName‘, ‘countryName‘, ‘qualificationZoneName‘, ‘stadiumName‘, and ‘teamName‘ as HasKey properties contributes to a robust and effective ontology design, enhancing clarity, efficiency, and reliability in knowledge representation and reasoning. Those data type properties are literals.

Moreover some data type properties like **cityName**, **countryName**, **foot:teamName**, **foot:stadiumName** and **foot:qualificationZoneName** are designated as "HasKey" attributes. This classification is attributed to their capability to uniquely identify teams, stadiums, countries, cities, and qualifications, respectively, contributing significantly to the dataset's integrity and efficiency in data retrieval and management.

Thanks to Protégé, we obtain the ontography which represents the way in which certain

individuals in the main classes interact with each other :

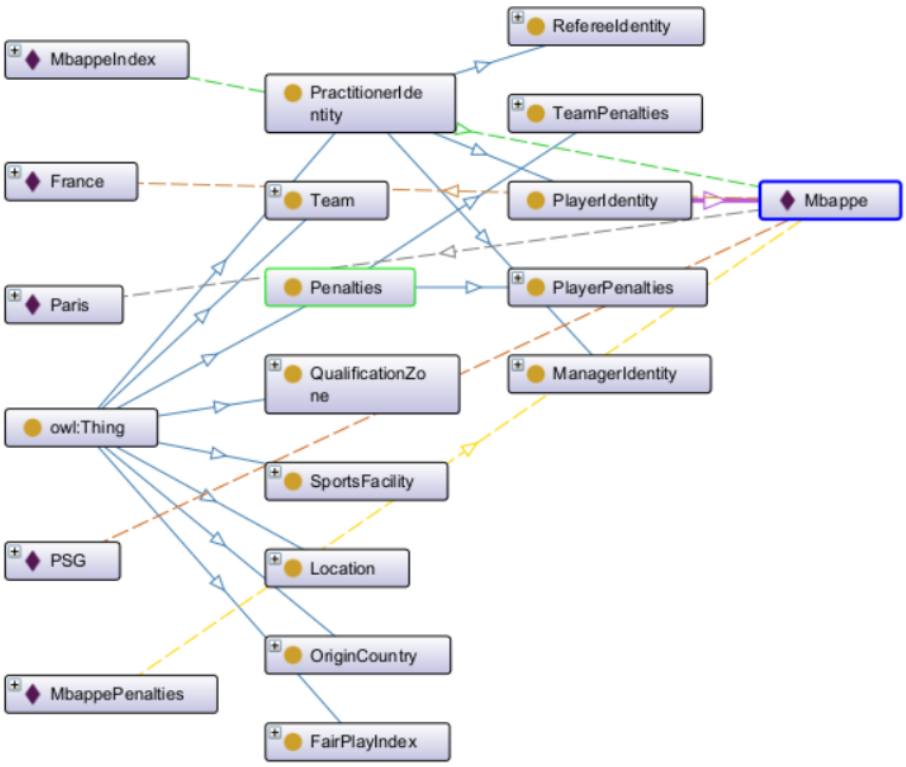

Figure 1: Ontography: Visualizing Interactions in Main Classes with Protégé.

We can express a minimum and maximum cardinality restriction to specify that a team must have a minimum of 18 players, and also specify that each manager has a minimum of 18 players.

3 Limitations Encountered with OWL 2 RL in Ontology Development

Despite its utility, OWL 2 RL poses several limitations when applied to ontology construction within our domain. These constraints hinder the precise modeling of concepts and relationships. Here are some common restrictions of OWL 2 RL in our domain, illustrated with examples:

Limitation of Axiom Expressiveness: OWL 2 RL lacks the capability to express certain types of complex axioms, thus hindering the representation of sophisticated relationships between entities. For example, articulating complex axioms like "A player cannot be a manager of the same team" or defining relationships such as "two players being teammates" may prove challenging. Another example: we cannot assert that "two players who play for different teams are opponents" or "two different clubs or national teams are opponents".

Limitation in Modeling Composition Operations: The absence of support for advanced composition operations, such as property intersections, in OWL 2 RL complicates the representation of certain complex relationships. For instance, establishing a relationship that links a player to a team through a management relation by a manager becomes challenging.

Fortunately, despite the constraints posed by OWL 2 RL, these challenges can be addressed through the utilization of Datalog. Datalog provides a means to articulate intricate rules beyond the scope of OWL 2 RL, offering greater flexibility in defining rules and relationships among entities. Additionally, Datalog empowers users to specify customized cardinality rules for entities and relationships, effectively overcoming constraints related to complexity and performance

inherent in cardinality restrictions within OWL 2 RL.
Some are illustrated as follows:

- Inferring player manager:

  $\text{foot:playsFor}(?p, ?t) \wedge \text{foot:manages}(?m, ?t) \rightarrow \text{foot:coaches}(?p, ?m)$

- Inferring attendsIn:

  $\text{foot:isIncludedIn}(?c, ?q) \wedge \text{foot:comesFrom}(?p, ?c) \rightarrow \text{foot:attendsIn}(?p, ?q)$

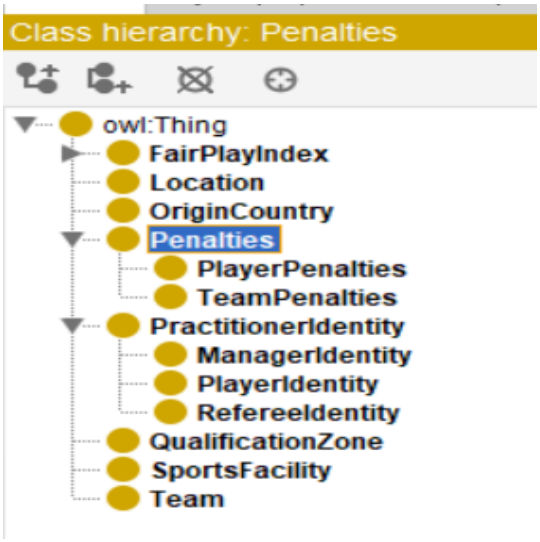

Figure 2: Ontology classes