Mixed Martial Arts Ontology developed in OWL and SWRL

Ali Khudiyev

November 2021

1 Introduction & Modelling

Mixed Martial Arts (MMA) is a full-contact combat sport based on striking, grapling and ground fighting. Top MMA organizations in the world are *Ultimate Fighting Championship (UFC)*, *Bellator MMA*, *Absolute Championship Akhmat (ACA)* which allow mixed-gender championships by separating male and female events. Such companies gain profit by selling fights as the end-user products each of which goes through the process of organization(i.e., fighter matching, place selection, pay-per-view prices, etc.) and evaluation(i.e., fight winner/drawer, fighter scoring). As in other combat sports, there are divisions among fighters participating in MMA. These divisions are to group fighters in the same weight class to make the championships fair since extra weights can be advantage and/or disadvantage depending on various factors. For example, UFC has the following weight classes[1]:

Weight class	**Weight**
Heavyweight	120.2 kg
Light heavyweight	102.1 kg
Middleweight	83.9 kg
Welterweight	77.1 kg
Lightweight	70.3 kg
Featherweight	65.8 kg
Bantamweight	61.2 kg
Flyweight	56.7 kg
Strawweight	52.5 kg

Table 1: UFC Weight Classes

Building an ontology is all about observing the relevant existing concepts and the relationships between them. For MMA ontology, there are whole bunch of concepts and relationships, however, I will illustrate the ones that are among the most important ones that make MMA what it is. The first concepts that come to the mind in MMA are the notion of organization and person. An MMA organization has a name and pays monthly salary to the organization workers. Organization workers can be CEO, judges, referees, organizers and fighters. Fighters participates at fights which is organized by organizers, controlled by referees and judged by judges. Judges gives points to both fighters based on which it is decided which fighter won/lost/drawed the fight. The fight takes place at some location and at a specific time, costs some amount of money which can be paid by a non organization worker to buy the fight. Every fight has a division that restricts how much the participating fighters weigh. Any two fighters have to have gender that is either male or female, in other words, (male vs female) match is impossible due to unfairness. The organization have certain age restrictions for its workers and also pays bonus money to a winning fighter. The whole information can be represented as the knowledge graph shown below:

[1] https://wayofmartialarts.com/ufc-weight-classes-divisions

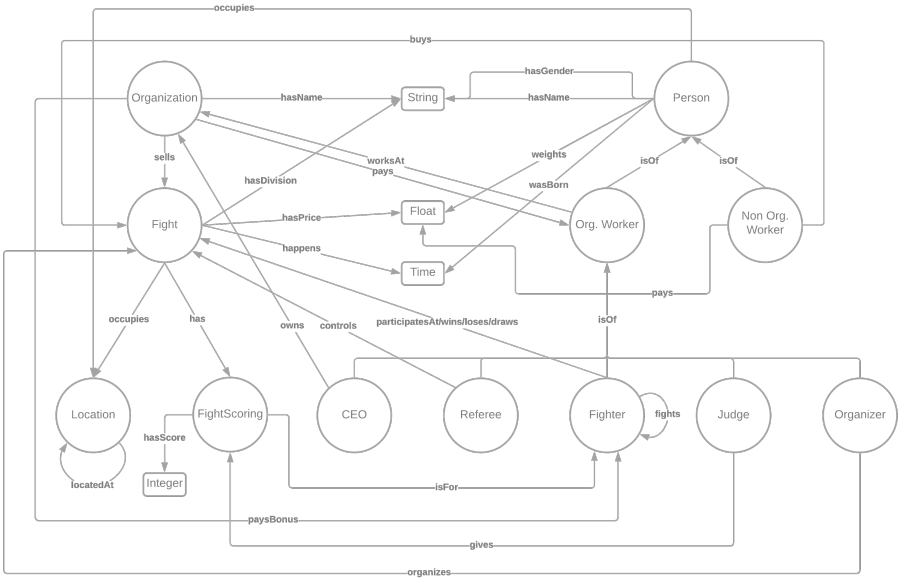

Figure 1: MMA Ontology Diagram

However, it is not enough to represent the whole just by using OWL alone and in fact, there are rules which represent some of the relationships hidden in the OWL graph. There are several rules that are among the most relevant ones as far as this project concerns and they are shown below in DL.

$$Fight \sqsubseteq = 2hasParticipant.Fighter \tag{1}$$

$$\begin{aligned} \text{Fight}(f) \wedge \text{Fighter}(x) \wedge \text{Fighter}(y) \wedge \text{participatesAt}(x, f) \\ \wedge \text{participatesAt}(z, x) \wedge \text{differentFrom}(x, y) \implies a = b \end{aligned} \tag{2}$$

$$\forall x \forall y(\text{Fighter}(x) \wedge \text{Fighter}(y) \wedge \text{wins}(x, y) \implies \text{loses}(y, x)) \tag{3}$$

$$\forall x \forall y(\text{Fighter}(x) \wedge \text{Fighter}(y) \wedge \text{draws}(x, y) \implies \text{draws}(y, x)) \tag{4}$$

$$\forall x(\text{Fighter}(x) \implies (\text{wasBorn}_x \geq 1970)) \tag{5}$$

$$\forall x(\text{Referee}(x) \implies (\text{wasBorn}_x \geq 1950)) \tag{6}$$

$$\forall x(\text{Judge}(x) \implies (\text{wasBorn}_x \geq 1950)) \tag{7}$$

Formula 1 indicates that any fight has to have exactly 2 participating fighters while 2 is to say that every couple of fighters who has fought against each other have to weigh the same which is due to the division rules of MMA. Formulas 7 are to restrict organization worker's ages, ?

2 Implementation in Protégé

The implementation goes through several stages: *creating class hierarchy and defining class equivalance(s)/ subsumtion(s)/ disjuntion(s)*, *defining relations*, *defining SWRL rules* and finally *creating individuals*. The class hierarchy is created by simply creating concepts used in the ontology and the class properties such as equivalance[2], subsumtion, disjuction between them are defined within the scope of the same classes. Relations are to link different concepts with their relevant predicates and have properties(i.e., reflexivity, transivity, etc.) as well. These 2 steps are what can be done using only OWL and they are carefully developed to avoid unintentional situations later on. The third step, which is defining SWRL rules, is developed by using SWRLTab in Protégé and tested with the OWL individuals that are created at the final development step of the ontology.

[2]Although equivalance is rarely used as it is mainly for merging two separately developed ontologies, there are classes which make it relevant to use it in this project.

2.1 Creating and Defining Classes

There are several concepts(classes) in this ontology:

- Organization
- Location
- Person
  - Organization Worker
    - CEO
    - Judge
    - Referee
    - Organizer
    - Fighter
  - Non Organization Worker
- Fight
- FightScore

$$\text{OrgWorker} \cup \text{NonOrgWorker} = \text{Person} \tag{8}$$

$$\text{OrgWorker} \cap \text{OrgWorker} = \emptyset \tag{9}$$

2.2 Defining Relations

A relation in logic has some arity, which is usually called k-ary relation. When k is 1, it is called unary and when k is 2, it is called binary relation. In Protégé, only binary relations are developed and therefore, we give off some level expressiviness as a trade-off with computation complexity. Due to such expressivity, any relation with more than arity 2 has to be reformulated with the help of additional concepts and relations. It is the same case with the classes *Fight*, *FightScore* and *Fighter* in this project; as it is impossible to have a 3-ary relation describing a *score* of *fighter* in a particular *fight*. There are also several properties of binary relations such as functional, symmetric, reflexive, transtive and so on. These properties are also used in the developement when it is relevant and necessary to avoid illogical KB and/or inferences.

Relation	**Domain**	**Range**	**Type**	**Description**
hasName	Thing	string	functional	Any concept has a name
sells	Organization	Fight	regular	Organization sells fights
buys	NonOrgWorker	Fight	regular	Non organization worker can buy fights
pays	NonOrgWorker	float	functional	Non organization worker pays to buy fights
participatesAt	Fighter	Fight	regular	Fighter participates at a fight by fighting
wins	Fighter	Fighter	inverse(loses)	Fighter can win another fighter
loses	Fighter	Fighter	inverse(wins)	Fighter can lose to another fighter
draws	Fighter	Fighter	regular	Fighters can draw

Table 2: Ontology Relations

2.3 Defining SWRL Rules

2.4 Creating Individuals

Individual	Asserted	Inferred
Org	Organization	
OrgCEO	Person	
OrgFighter1	Fighter	
OrgFighter2	Fighter	
OrgFighter3	Fighter	
OrgFighter4	Fighter	
Fight1	Fight	
Fight2	Fight	
FightScore1_1	FighScore	
FightScore1_2	FighScore	
FightScore2_1	FighScore	
FightScore2_2	FighScore	

Table 3: Individuals

2.5 SPARQL Queries

...

3 Reasoning with Pellet

...

4 Evaluation & Conclusion

OWL and SWRL help us to create ontologies with certain subset of first order logic that makes the inference decidable. To do so, it sacrifices some sort of expressiviness that could be otherwise used to develop certain ontologies(e.g. an ontology that requires creation of new individuals under some conditions). Another type of problem may arise when trying to represent object states that is dynamic over time since there is no concept of time in OWL and SWRL. Consider we want to simulate *Conway's Game of Life*[3] by using SWRL rules. Since there are cells that are either alive or dead depending on the state of the neighbours, it is not very intuitive to represent a cell which can transform from being dead to alive or vice versa as iterations are developed. However, we could still simulate the GoL by representing the concept of time/iteration as an extra dimension. After thinking on the proper way of representing time, I settled in the following representation: *each individual cell has its own (current) time or iteration rather than a global notion of time*. This decision comes from the realization that rules can be applied randomly when relevant which interrupts the synchronization between cell states and the iteration that they are in. So, rather than having a global notion of iteration, each cell has its own sense of iteration which is developed as the reasoner processes or does some transformation upon the cell.

$$\begin{gathered}\text{Time}(T)\\ \text{AliveCell}(c1)\\ \text{DeadCell}(c2)\\ \text{DeadCell}(c3)\\ \text{AliveCell}(c4)\\ \text{Time}(T) \wedge \text{hasValue}(T,t) \implies \text{hasValue}(T,t+1)\end{gathered}$$

Figure 2: Notion of global time

[3] https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

$$Cell(c1)$$
$$Cell(c2)$$
$$Cell(c3)$$
$$Cell(c4)$$
$$isLiveAt(c1, 1)$$
$$isLiveAt(c2, -1)$$
$$isLiveAt(c3, -1)$$
$$isLiveAt(c4, 1)$$
$$\text{Cell}(c) \wedge \text{isLiveAt}(c, t) \wedge \text{hasSufficientNeighbours}(c, t) \wedge \text{isNegative}(t) \implies \text{isLiveAt}(c, 1 - t)$$
$$\text{Cell}(c) \wedge \text{isLiveAt}(c, t) \wedge \text{hasSufficientNeighbours}(c, t) \wedge \text{isPositive}(t) \implies \text{isLiveAt}(c, 1 + t)$$
$$\text{Cell}(c) \wedge \text{isLiveAt}(c, t) \wedge \text{hasInsufficientNeighbours}(c, t) \wedge \text{isNegative}(t) \implies \text{isLiveAt}(c, -1 + t)$$
$$\text{Cell}(c) \wedge \text{isLiveAt}(c, t) \wedge \text{hasInsufficientNeighbours}(c, t) \wedge \text{isPositive}(t) \implies \text{isLiveAt}(c, -1 - t)$$
$$\text{Cell}(c) \wedge \text{hasNeighbour}(c, c1) \wedge \text{isLiveAt}(c1, t) \wedge \cdots \wedge \text{isPositive}(t) \implies \text{hasSufficientNeighbours}(c, t)$$
$$\text{Cell}(c) \wedge \text{hasNeighbour}(c, c1) \wedge \text{isLiveAt}(c1, t) \wedge \cdots \wedge \text{isPositive}(t) \implies \text{hasInsufficientNeighbours}(c, t)$$

Figure 3: Notion of local time

The predicate isLiveAt(cell , t) represents an alive cell when $t > 0$ and a dead cell when $t < 0$. The analogy here is that time is bidirectional; positive time represents the time in our world and negative time represents the time in afterlife. So, isLiveAt(cell , 3) indicates that the cell is alive at the iteration 3 and isLiveAt(cell , -4) means that the cell is alive at the iteration 4 in afterlife, in other words, the cell is dead at iteration 4 (in our world). A cell is alive in our world when it is dead in the afterlife and vice versa. This representation avoids the creation of new cells (as it is also prohibited in DL) and helps us to illustrate the notion of iteration/time. However, the above shown rules are not the only ones required to build such interesting ontology and the missing pieces that I have not intentionally mentioned is the formulas describing hasSufficientNeighbours(cell, t) and hasInsufficientNeighbours(cell, t); these are also updates through the process. In fact, hasSufficientNeighbours(cell , t) is updated to indicate that if cell has sufficient number of neighbours at iteration t to keep living (if live before) or turn to an alive cell (if dead before) at the consequent$(t + 1)$ iteration. It is the opposite case for hasInsufficientNeighbours(cell , t) which is updated to indicate whether a cell has less or more than enough live neighbours at the iteration t.