An Ontological Modelling of Prototype Theories

Daniele Porello[1], Guendalina Righetti[2], Nicolas Troquard[2], Roberto Confalonieri[3] and Oliver Kutz[2]

[1]University of Genoa, via Balbi 4, 16126, Genova, Italy.
[2]Free University of Bozen-Bolzano, piazza Domenicani 3, 39100, Bolzano, Italy
[3]University of Padua, Via Trieste 63, I-35121, Padova, Italy

Abstract

Prototype theories are an important family of cognitive theories of concepts that model the classification under a concept in terms of the proximity of an object to the prototype of the concept. While logic-based definitions of concepts are standard in Description Logics and OWL, prototype-based definitions are not directly available, although they are very useful whenever we need to model commonsense concepts or data dependent classifications.

We propose a strategy to define prototypes in OWL enriched with SWRL (Semantic Web Rule Language). By means of data properties we model the weighted features of prototypes, and, by using SWRL constraints, we implement the computation of the proximity of an instance to a prototype. We also leverage a foundational ontology to provide semantics of the features occurring in prototype descriptions. We exemplify our treatment in Protégé.

Keywords

Prototype theories, OWL, Semantic Web Rule Language (SWRL), Foundational Ontology, DOLCE, Protégé

1. Introduction

Various cognitive theories have been proposed in the literature to define and understand the representation of concepts [1]. One prominent theory among these is prototype theory, which has received significant formal and empirical development [2, 3, 4]. As a result, it has served as a foundational inspiration for numerous formal systems that aim to provide improved models of human concepts in fields such as Logic and Knowledge Representation [5, 6, 7, 8].

Prototype theory traces its origins back to Wittgenstein's concept of family resemblance [9], but its empirical foundation is due to the findings of Eleanor Rosch [2]. Her experiments showed that many concepts exhibit typicality effects (namely, some class members are more representative than others), and that members and non-members of a category form a continuum. Since then, prototype theory is mostly understood as providing a *summary representation* [1], representing the central tendency of a category as a whole. Although there exist several variants of prototype theory, their central core is that it is possible to represent concepts as a weighted list of the attributes most commonly found among concept instances. The more attributes an individual satisfies, the more similar it will be to the prototype, and the more typical it will be for the category.

To formally capture this idea, Masolo and Porello [5] propose a logical rendering of prototypes by introducing weights into the language of first-order logic. The logic-based rendering of prototypes paves the way towards introducing semantic constraints to make the meaning of the features or the attributes explicit. Porello et al. [6] translate this approach into Description Logic (DL) by introducing novel operators into the syntax of standard DL languages. These operators take lists of concept descriptions, associate a weight to each of them, and return complex concepts which apply to the instances of the domain that satisfy a sufficient number of the concept descriptions. A threshold is set to decide what is deemed to be sufficient, and a new *value function* is introduced to define the semantics. This function sums up the weights of the features an instance satisfies and returns a number which allows expressing instances' typicality. When the number reaches the threshold, the instance is classified as a member of the concept.

Threshold concepts for representing prototypes have also been used by Baader and Ecke [7], who propose however a different interpretation. The semantics makes use of a *prototype distance function* defined over weighted

9th Workshop on Formal and Cognitive Reasoning, September 26, 2023, Berlin, Germany
daniele.porello@unige.it (D. Porello);
Guendalina.Righetti@stud-inf.unibz.it (G. Righetti);
nicolas.troquard@unibz.it (N. Troquard);
roberto.confalonieri@unipd.it (R. Confalonieri);
oliver.kutz@unibz.it (O. Kutz)
https://danieleporello.net (D. Porello);
https://www.unibz.it/it/faculties/engineering/academic-staff/person/40334-guendalina-righetti (G. Righetti);
https://www.inf.unibz.it/~ntroquard/ (N. Troquard);
https://www.math.unipd.it/~confa/ (R. Confalonieri);
http://www.inf.unibz.it/~okutz/ (O. Kutz)
0000-0003-3655-0218 (D. Porello); 0000-0002-4027-5434 (G. Righetti); 0000-0002-5763-6080 (N. Troquard); 0000-0003-0936-2123 (R. Confalonieri); 0000-0003-1517-7354 (O. Kutz)

alternating parity tree automata, which assign to each element of an interpretation a distance value expressing its similarity to the prototype.

Relatedly, typicality operators [10, 11, 8] have also been proposed in the DL literature to reason non-monotonically about the typicality of instances. In these cases, it is possible to identify the most typical members of a concept by assuming a preference order over the elements of the domain of interpretation: depending on the system, either the individuals further down or the ones higher up in the order are deemed as the more typical instances of a concept. Moreover, prototypes and typicality in ontologies have also been approached in [12, 13].

In this paper, we propose to model prototypes in OWL 2 using SWRL rules [14].[1] The Semantic Web Rule Language (SWRL) is an extension of OWL (Web Ontology Language), the most widespread language for authoring ontologies. SWRL allows for the specification of Horn-like rules within an OWL ontology.

SWRL is particularly suitable for our purposes because it combines OWL class and property expressions with built-in relations, such as equality, inequality, and arithmetic operations. These will be instrumental for the representation of prototypes within the ontology and to compute the similarity measure that enables classifications, cf. [15, 16].

After discussing three variants of prototype theories with their measures of similarity, we present our proposal to model prototypes by means of data properties and SWRL constraints. To ascribe meanings to features or attributes occurring in prototype descriptions, and to enable reasoning with them, we introduce a background foundational ontology (in particular, we use DOLCE, cf. [17, 18]). Moreover, we exemplify our treatment by means of Protégé, using the plug-in for SWRL found at https://github.com/protegeproject/swrlapi/wiki.

The main novelty of our approach, in comparison to the previously mentioned related work, lies in the combination of three key ingredients: *i*) the use of SWRL to implement computations of similarities, *ii*) the use of a foundational background ontology to provide semantics of features, and *iii*) the use of threshold concepts to interface prototypes with ontologically crisp information. The threshold mechanism allows for importing prototype-based classifications without altering the semantics or the reasoning services of OWL.

The remainder of this paper is organised as follows. In Section 2, we overview prototype theories. In Section 3, after a brief overview of the ontology DOLCE, we discuss how to represent the attributes and the attribute-values of a prototype in terms of qualities and quality structures of DOLCE. Section 4 presents a general strategy for representing prototypes in OWL 2 plus SWRL and for computing similarity by means of SWRL constraints. Section 5 combines the SWRL rendering with the background ontology based on DOLCE. Section 6 exemplifies our approach. Section 7 discusses how to learn the weights and the diagnosticity values from a data set. Finally, Section 8 concludes and indicates future work.

[1] See also https://www.w3.org/Submission/SWRL/.

2. Prototype theories

Prototype theories of concepts are models proposed to explain human conceptualisations and classification of objects under concepts [2, 1, 19, 4].

Accordingly, a concept is defined by a *prototype*, a summary, abstract representation that captures the central tendency of a category, namely the most typical instances or objects belonging to the class. Classification is not always deemed to be crisp: the degree of classification of an object under a concept is instead rendered by means of proximity, or similarity, of the object to the prototype.

Notwithstanding this common core, there exist several variants of prototype theories.

According to Rosch and Mervis [2], a prototype is an (unstructured) list of the features or attributes usually found across the instances of the concept we are considering. Each feature is assigned a weight to indicate its importance in describing the concept. These weights are determined based on the frequency distribution among exemplars of the concept, where features more frequently observed receive higher weights.

Suppose that the features or attributes that describe the prototype are given by $\mathbf{Q} = \{q_1, \ldots, q_n\}$. The prototype for a concept is then represented by the following set of pairs:

$$\{(q_1, w_1), \ldots, (q_n, w_n)\}$$

The similarity of an exemplar a to a concept is measured through its degree of *family resemblance*, which can simply be computed by using the following sum, where the $w_{l_1}, \ldots, w_{l_m}$ are the weights in $\{w_1, \ldots, w_n\}$ associated to prototypical features that are also exhibited by object a.

$$S_a = \sum_{j \in \{l_1, \ldots, l_m\}} w_j \tag{1}$$

Hampton's version of prototype theory is similar to the one originally proposed by Rosch and Mervis, see [4]. However, Hampton suggests a slightly more elaborate way to compute the similarity between objects and prototypes, which also takes into account the degree to which an object exhibits a feature.

Accordingly, the similarity between objects and prototypes is now computed by the following formula:

$$S_a = \sum_{i=1}^{n} (w_i \times v_{(i,a)}) \quad (2)$$

According to Hampton, w_i and $v_{(i,a)}$ range as follows: $0 \leq w_i \leq 1$ and indicates the importance of the i-th feature of the prototype for the classification, and $-1 \leq v_{(i,a)} \leq 1$ represents the degree to which the instance a has q_i, see [4].

A more structured representation of prototypes is proposed by Smith and colleagues. In [3], a prototype is defined by three ingredients, namely:
i) an *attribute-value* structure, that is, a set of attributes of objects associated with values (e.g. colour-red, weight-heavy, shape-round),
ii) an indication of the *salience* of the attribute values,
iii) an indication of the *diagnosticity* of the attributes for the classification under the concept.

In [3], the salience value of an attribute-value (e.g. colour-red) captures "the subjective frequency with which the value occurs in instances of the concepts, and the perceptibility of the value" (p. 489). Salience values are introduced to account for attribute-values that are closely related to the concept. E.g., people are faster at establishing that apples are red than apples are round, suggesting that red is more salient than round in the prototype for apple, cf. [3].

By contrast, diagnosticity values measure how useful the attribute is in separating instances of the concept from instances of contrasting concepts.

In this setting, a representation of a prototype can be given by means of a set of weighted attribute-values. Suppose that the attributes of interest are $\mathbf{A} = \{A_1, \ldots, A_n\}$. Moreover, for each attribute A_j, assume that the possible values of attribute A_j are $\mathbf{Q}_{A_j} = \{q^1_{A_j}, \ldots, q^{n_j}_{A_j}\}$. We indicate by d_{A_j} the *diagnosticity* of some attributes A_j. By $s^i_{A_j}$, we denote the *salience* of the i-th value of attribute A_j. A prototype can then be specified by the following set of triples, for some $A_j \in \mathbf{A}$ and $q^{i_1}_{A_j}, \ldots, q^{i_l}_{A_j} \in \mathbf{Q}_{A_j}$ (cf. [3], Figure 1).

$$P = \{(q^{i_1}_{A_j}, d_{A_j}, s^{i_1}_{A_j}), \ldots, (q^{i_l}_{A_j}, d_{A_j}, s^{i_l}_{A_j})\} \quad (3)$$

In this setting, the similarity between an object and the prototype is computed employing Tversky's contrast rule (see [3]). To employ this rule, the authors also represent objects, or instances, as descriptions in terms of sets of weighted attribute-values. Object descriptions are sets of pairs of attribute-values and salience values (while diagnosticity specifically operates for prototypes). For some $A_j \in \mathbf{A}$ and $q^{i_1}_{A_j}, \ldots, q^{i_l}_{A_j} \in \mathbf{Q}_{A_j}$, an instance I is described as follows:

$$I = \{(q^{i_1}_{A_j}, s^{i}_{A_j}), \ldots, (q^{i}_{A_j}, s^{i_l}_{A_j})\} \quad (4)$$

For example, restricting to two attributes, a red apple is represented by the description $\{(\textsc{Red}, s^a_{\textsc{Color}}), (\textsc{Round}, s^a_{\textsc{Shape}})\}$, where $s^a_{\textsc{Color}}$ (resp. $s^a_{\textsc{Shape}}$) is "the number of votes" in favour of being Red (resp. Round), e.g. the number of instances that are deemed to be red, which in principle could be different from the salience established by the prototype (cf. [3]) .

To compute Tversky's contrast rule, we define, for each attribute A_j, three sets of values (in $\mathbb{N}$), cf. [3], p. 491. For each attribute-value $q^i_{A_j}$, PI_{A_j} lists the number of votes about $q^i_{A_j}$ that are common to P and I; $P\bar{I}_{A_j}$ lists the number of votes that are proper to P and not to I, while $\bar{P}I_{A_j}$ lists the number of votes proper to I and not to P.

$$PI_{A_j} = \{s \mid (q^i_{A_j}, s') \in P, (q^i_{A_j}, s'') \in I \text{ and } s = min(s', s'')\}$$

$$P\bar{I}_{A_j} = \{s \mid (q^i_{A_j}, s') \in P, (q^i_{A_j}, s'') \in I \text{ and } s = s' - s'', \text{ if } s \in \mathbb{N}, 0 \text{ otherwise}\}$$

$$\bar{P}I_{A_j} = \{s \mid (q^i_{A_j}, s') \in P, (q^i_{A_j}, s'') \in I \text{ and } s = s'' - s' \text{ if } s \in \mathbb{N}, 0 \text{ otherwise}\}$$

The similarity between an instance a and a prototype is then computed by the Tversky rule:

$$S_a = \sum_{A_j \in \mathbf{A}} \{d_{A_j} \times (\sum PI_{A_j} - \sum P\bar{I}_{A_j} - \sum \bar{P}I_{A_j})\} \quad (5)$$

Intuitively, Tversky's measure evaluates similarity by taking into account commonalities and differences between the description of the prototype P and the description of the instance I.

We sketch an example discussed in [3]. We restrict ourselves to two attributes, Color and Shape, with values of color in $\{\textsc{Red}, \textsc{Brown}\}$ and values for shapes in $\{\textsc{Round}, \textsc{Squared}\}$. A prototype for the concept Apple includes then the following triples:

$$(\textsc{Red}, d_{\textsc{Color}}, s^{\textsc{Red}}_{\textsc{Color}}), (\textsc{Brown}, d_{\textsc{Color}}, s^{\textsc{Brown}}_{\textsc{Color}})$$
$$(\textsc{Round}, d_{\textsc{Shape}}, s^{\textsc{Round}}_{\textsc{Shape}}), (\textsc{Squared}, d_{\textsc{Shape}}, s^{\textsc{Squared}}_{\textsc{Shape}})$$

According to [3], the diagnosticity of the attribute Color for classifying apples is higher than the diagnosticity of the attribute Shapes, while the salience of the value Red for apples is higher than the salience of Brown.

The similarity measures that we have presented define a value for the proximity of an instance to the prototype. As such, classification under a concept is graded. To obtain the set of entities that falls under a concept (i.e. the *extension* of the concept) by means of the prototype mechanism of classification, a *threshold* t is set, cf. [4].

Definition 1. *Given a threshold t, an object a belongs to the extension of the concept C if and only if*

$$S_a \geq t. \tag{6}$$

We shall use the threshold mechanism to introduce a class of the ontology, the class of entities that are classified by the concept.

We conclude this section by noticing that a prototype may include complex features that are not reduced to attribute value pairs. For instance, complex properties such as "has legs" or "you can drive it" are used by [2] in the summary description. We shall use the word *features* when we wish to emphasize general properties that may occur in the description of a prototype.

3. Ontological representation of prototypes

In this section, we discuss how to represent the features and the concept defined by the prototype within an ontological setting. Then, in the next section, we will approach how to render prototypes in OWL 2 plus SWRL.

By embedding prototypes within a rich ontological framework, we can provide a formal representation of the semantics of the features involved in the description of the prototype and this, in turn, enables reasoning about the content of those features.

We place our discussion within the foundational ontology DOLCE (*Descriptive Ontology for Linguistic and Cognitive Engineering*), see [17, 18]. DOLCE is a cognitive-oriented foundational ontology, it aims to represent the conceptualisation of a cognitive agent, rather than the agent-independent structure of the world. Its cognitive inspiration is shown by the adoption of cognitively founded theories as a basis for many of its distinctions, viz. the adoption of quality structures inspired by the Conceptual Spaces [20].

DOLCE partitions the elements of the domain of discourse (the *particulars*) into four main categories: *Endurants*, i.e. objects, *Perdurants*, i.e. events, *Qualities* and *Abstract*. The latter two classes deserve our attention here, as we focus on the representation of individual qualities and *qualia*, see [18].

Qualities are properties of objects that can be perceived or measured. Qualities are dependent entities, in that they are strictly attached to their objects, their *bearers* (they *inhere* in their object). For this reason, DOLCE terms them individual qualities. The color of *this* rose is never the same entity as the colour of *that* rose, as they are attached to different objects. To compare different individual qualities, DOLCE introduces *qualia*, i.e. values of individual qualities at a certain time[2]. *Qualia* are intended as positions occupied by an individual quality into a quality space, which is modelled as an abstract region, i.e. a subclass of *Abstract*. Qualities are also divided into types, such as color, weight, shape, etc. qualities. The quality spaces associated to each quality type are inspired by Gärdenfors quality dimensions, cf. [20]. We illustrate the mechanism of ascriptions of qualities and qualia in DOLCE by means of the following example, visualised in Figure 1.

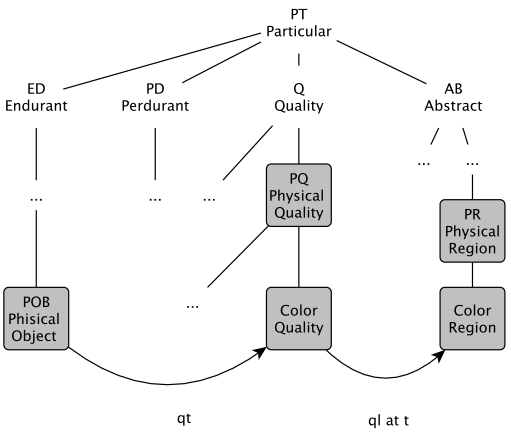

Figure 1: Fragment of the DOLCE taxonomy and relations for quality and qualia.

Given a particular apple a (i.e. an instance of the class POB of physical objects in DOLCE), the relation qt (spelled "has quality") associates the (color) individual quality, say c_a, an element of the class of color qualities. The relation ql ("quale of at t") associates to c_a its *quale*, i.e. an element of a quality space, a color region, e.g. red.[3] Thus, in DOLCE, the ascription of a *quale* to the individual quality of an entity is done by adding the following two statements (facts) to the ontological theory: $\mathsf{qt}(a, c_a)$ and $\mathsf{ql}(c_a, \mathsf{red}, t)$. The time parameter of the ql relation allows for modelling change of qualia in time.

Therefore, the attribute-values of prototype theory can be represented in DOLCE by means of the objects, the individual qualities of the object, and the *qualia* of the quality, i.e. the corresponding quality regions.

The ascription of an attribute value to an entity is then rendered via the ql and qt relations. For an axiomatisation of these relations in DOLCE, we refer to [18].

[2]The use of the term *quale* here is technical to DOLCE, i.e. we do not commit to embracing neither a subjective nor an intersubjective view of qualia. Notice however that in prototype theories qualia have to be measurable and comparable.

[3]In Figure 1,ql is depicted as a binary relation, to simplify. In DOLCE, ql is ternary, it takes also a time argument t, which is in DOLCE an element of the abstract region of time.

To use the representation of quality and quality structures of DOLCE in OWL, we shall simplify it a little.[4] We view the ontology in OWL as providing a snapshot at given time of the information about a domain of objects. So, we omit the time argument in ql, which remains implicit. The reason behind this choice is to avoid dealing with ternary relations in OWL, which makes the rendering more complex. Thus, $\mathsf{ql}(x, y)$ is now defined as a binary relation, it relates an individual quality x to a *quale* y, while the time of the association remains implicit. We can also introduce a defined relation hasQuale, which is the composition of ql and qt. For instance, $\mathsf{hasQuale}(a, \mathsf{red})$ is the composition of the relations $\mathsf{qt}(a, c)$ and $\mathsf{ql}(c, \mathsf{red})$. We shall use these relations to simplify the rendering in OWL.

As usual in ontological modelling, DOLCE provides binary classifications of objects under properties and relations; as such, an entity has or has not a certain *quale*. By contrast, prototype theories use graded attribute values, those measured by the function $v_{(i,a)}$, e.g. in Equation 2. Thus, to enable the ontology to acquire the bits of information about the quale of an entity, we need a threshold to state that if $v_{(i,a)}$ is "large enough" (e.g. $v_{(i,a)} = 1$), then the entity a has the quale indicated by the attribute value q_i. We shall see how to render this aspect in the next section.

As we mentioned, prototypes may include general properties, besides attribute-values. The ontological rendering of such properties is a matter for a dedicated analysis. E.g. "has legs" may be represented by using the "part of" relation, where the semantics of such a relation is captured by a mereological theory, which is included in DOLCE. We refer to [21] for the strategies for representing ontologically the information conveyed by the features of a prototype.

The contribution of a foundational ontology such as DOLCE to our approach is intended to provide the semantics to reason about qualities and qualia – and more generally about features – and to interface the classifications provided by means of prototypes with background information about the domain. For instance, it is by means of the information provided by the ontology that we can infer that an entity cannot be wholly red and brown at the same time, or that it cannot have both a round and a squared shape.

4. Computing similarity in SWRL

We illustrate the general mechanism for representing in SWRL the computations required by the prototype theories. In the next section, then, we discuss the integration with a background ontology and we exemplify the strategy in Protégé on a concrete case in Section 6.

We assume that the information provided by a prototype is an input of our representation, that is, we assume to have: the set of attributes or features, $\mathbf{F} = \{F_1, \ldots, F_n\}$, their values, $\mathbf{Q}_{F_j} = \{q^1_{F_j}, \ldots, q^{n_j}_{F_j}\}$, the diagnosticity d_{F_j} of each features F_j, the salience $s^i_{F_j}$ of each value, and the values $v_{(j,a)}$, the degree to which each entity a has feature value $q^i_{F_j}$.[5]

We sometimes use the labels F_j, for general features, not to restrict to attribute values, according to [4], while we use labels A_j to intend attributes as quality structures.

We remind the definition of SWRL atoms and rules, cf. [14]. The syntax of SWRL extends the usual syntax of OWL by adding (individual) i-variables (as we shall see, they range over the individuals of the domain) and (data) d-variables (which range over data values).

An i-object is an individual name or an individual variable, a d-object is a data literal or a data variable. Atoms are then: i) applications of concept description to i-objects, ii) applications of an object property to a pair of i-objects, iii) the application of a data property to a pair consisting of an i-object and a d-object. Variables are denoted by `?x`. SWRL rules consist of an implication between an antecedent (the body) and a consequent (the head). The antecedent and the consequent contain sets of atoms, interpreted conjunctively. The antecedent provides the conditions that need to be satisfied for the rule to be applicable. The consequent specifies the logical consequences that must hold when the antecedent is satisfied. For an exhaustive overview of SWRL syntax and semantics, we refer to [14]. A SWRL rule has the following form, where `a1`, ..., `an` and `b1`, ..., `bn` are atoms, `a1`, ..., `an` is the antecedent (the body), and `b1`, .., `bn` is the consequent (the head):

```
a1 ^ ... ^ an -> b1 ^ ... ^ bn
```

To import the information required by the prototype theory, we shall introduce number of data properties. To make the domain of such properties semantically meaningful, we shall leverage a background ontology, however we postpone this discussion to Section 5. For now, we implicitly assume that they associate numbers to objects of the ontology.

4.1. Tversky similarity

We start by representing the Tversky formula, cf. Equation 5, which is the most involving case. To do that, we assume that the prototype is *reified*, i.e. that there is an individual name p to which we can ascribe certain attribute-values. In this setting, as we discussed in

[4] For OWL versions of DOLCE, cf. http://www.loa.istc.cnr.it/index.php/dolce/

[5] These numbers come from experimental data and statistical inference in prototype theories. We discuss how to retrieve them by means of learning algorithms in Section 7.

Section 2, a prototype is described by salience numbers for attribute-values and by diagnosticity numbers of attributes. This information can be rendered by means of the following instantiated data properties (intended as facts of the ontology), for each attribute A_j, attribute-value $q^i_{A_j}$, salience $s^i_{A_j}$, of the prototype p.

$$\{\texttt{hasAJQISal(p,sij)}, \texttt{hasAJDia(p,dj)}\}$$

`hasAJDia` associates to objects the number d_j (the diagnosticity of the attribute A_j), while `hasAJQISal` associates the salience value $s^i_{A_j}$ (the salience of the i-th value of attribute j).

In this setting, an instance a is also represented by a set of data properties, which associate the salience values.

$$\{\texttt{hasAJQISal(a,sij)}\}$$

To represent Equation (5) in SWRL, we need to compute the values of $\sum PI_{A_j}$, $\sum P\bar{I}_{A_j}$, and $\sum \bar{P}I_{A_j}$.

To do that, we introduce the following new data properties.

`hasPI-AJQI` associates the salience for common attribute values $q^i_{A_j}$ in PI_{A_j}
`hasPI-AJ` associates the value of $\sum PI_{A_j}$.
`hasPI'-AJQI` associates the values for $q^i_{A_j}$ in $P\bar{I}_{A_j}$.
`hasPI'-AJ` associates the value of $\sum P\bar{I}_{A_j}$.
`hasP'I-AJQI` associates the values for $q^i_{A_j}$ in $\bar{P}I_{A_j}$.
`hasP'I-AJ` associates the value of $\sum \bar{P}I_{A_j}$.

To compute the values of those data properties, we introduce the following SWRL constraints.

```
hasAJQISal(a,?s') ^ hasAJQISal(p,?s'')
^ minimum(?s,?s',?s'')
->  hasPI-AJQI(a,?s)

hasPI-AJQ1(a,?s1) ^...^ hasPI-AJQM(a,?sm)
^ swrl:add(?s,?s1,...,?sm) ->  hasPI-AJ(a,?s)

hasAJQISal(a,?s') ^ hasAJQISal(p,?s'')
^ swrl:subtract(?s,?s',?s'')
^ swrl:greaterThanOrEqual(?s,0)
->  hasP'I-AJQI(a,?s)

hasAJQISal(a,?s') ^ hasAJQISal(p,?s'')
^ swrl:subtract(?s,?s',?s'')
^ swrl:lessThan(?s,0)
->  hasP'I-AJQI(a,0)

hasP'I-AJQ1(a,?s1) ^...^ hasP'I-AJQM(a,?sm)
^ swrl:add(?s,?s1,...,?sm) ->  hasP'I-AJ(a,?s)
```

Analogous rules are set for `hasPI'-AJ`. The construct `minimum` is true when s is the minimum of s' and s''.[6] Analogously, `swrl:subtract`, `swrl:add`, `swrl:greaterThanOrEqual`, `swrl:equal` implement subtraction, addition, $\geq$ and $=$ (respectively).

Given the values of `hasPI-AJ(a,?s)`, `hasP'I-AJ(a,?s)`, and `hasPI'-AJ(a,?s)`, we can compute the full value of salience of an attribute by `hasAJSal`, as follows.

```
hasPI-AJ(a,?s1) ^ hasP'I-AJ(a,?s2)
^ hasPI'-AJ(a,?s2)
^ swrl:add(?s,?s1,-?s2,-?s3) -> hasAJSal(a, ?s)
```

To simplify the presentation, we are using `-?si` to indicate the inverse of the value of `?si`, instead of adding a further rule.

Then, we multiply the value of `hasAJSal` by the diagnosticity of A_j, to obtain `hasAJVal`.

```
hasAJDia(a,?d) ^ hasAJSal(a,?s)
^ swrl:multiply(?v,?d,?s)  -> hasAJVal(a,?v)
```

Finally, the Tversky similarity, represented by the data property `hasTVal`, is computed by the following rule:

```
 hasA1Val(a,?v1) ^ ... ^ hasAMVal(a,?vm)
^ swrl:add(?t,?v1,...?vm) -> hasTVal(a,?t)
```

4.2. Hampton's similarity

We discuss now how to implement Hampton's measure, cf. Equation (2). We introduce the data properties `hasQJWei` to introduce the weights of each feature value and `hasQJDeg` to set, for each instance a, the value $v_{(j,a)}$. The data property `hasQJVal` serves to compute the products of weights and degrees, by means of the following rule.

```
hasQJDeg(?x,?d) ^ hasQJWei(?x,?w)
^ swrlb:multiply(?v,?d,?w)
->  hasQJVal(?x,?v)
```

To compute the similarity to the prototype according to Equation (2), we introduce the data property `hasHVal` which associates Hampton's proximity value of an instances to the prototype.

Then, we compute its value by means of the following SWRL constraint:

```
hasQ1Val(?x,?v1) ^ ... ^ hasQMVal(?x,?vm)
^ swrlb:add(?v,?v1,...,?vm)
-> hasHVal(?x,?v)
```

Notice that the case of $v_{(j,a)} \in \{0, 1\}$ corresponds quite closely to the threshold operators defined in [6, 22].

[6] See documentation here https://protegewiki.stanford.edu/images/5/57/SWRL-IQ_manual.pdf

5. Ontological background for prototypes

In the previous section, we introduced a number of data properties to associate the values required by the computations of similarities. Those data properties are simply devices to implement by SWRL the prototype-based classifications. That is, at that level, they are not attached to any ontological, or semantic, information.

To link the features and attributes occurring in prototype descriptions to their ontological understanding, we proceed as follows. We present the approach for features that are construed as attribute-values, which can be represented ontologically in terms of qualities and qualia. Each type of feature, e.g. parthood, would in principle require a dedicated ontological analysis, see [21].

We define the background ontology to be an excerpt of DOLCE, see Section 3. Accordingly, we partition the class `PhysicalRegion` into the types of attributes that we are considering (e.g. `ColorRegion`, `ShapeRegion`, etc.). The elements of those classes are ontologically *qualia*, corresponding to the attribute-values listed in the prototype description. Thus, given attribute types $\mathbf{A} = \{A_1, \ldots, A_n\}$, `PhysicalRegion` is partitioned into n disjoint classes of attribute-values.

In usual ontological modelling, classifications are binary, an entity has or has not a certain attribute-value (an entity is red or it is not). To embed soft information into the discrete world of ontologies, we can again exploit a thresholding mechanism. We introduce the binary relations (i.e. the object properties) `hasAJQuale`, one for each attribute type, which relate entities to *qualia* of the specific type, see Section 3.[7]

More specifically, the domain of `hasAJQuale` is POB (physical objects) and the range is the j-th element of the partition of `PhysicalRegion` A_j. The `hasAJQuale` relations are set as functional, an object has only one value for each attribute type (at a time).

When we allow for degrees of ascriptions of attribute-values, we have to decide a threshold for inferring that an entity has a quale q_j, whenever the entity has it up to a certain degree. A very cautious way of setting this threshold is expressed by the following SWRL constraint:

```
hasQJDeg(?x,1) -> hasAJQuale(?x,qj)
```

By means of this constraint, we can populate the object property `hasAJQuale` of all those pairs (of entities and qualia) for which $v_{(i,a)} = 1$. Then, we can exploit the axioms of the ontology to reason about attribute-values ascriptions. For instance, the reasoner shall exclude that objects might have two qualia of the same type.

In Smith and Osherson [3], statements `hasAJQuale(?x,qij)` are bivalent, therefore the inputs of the prototype mechanism can be retrieved by means of qualia ascriptions. Once `hasAJQuale` is instantiated for certain a in POB and its quale q, the following rules populate the data properties by ascribing to a the pertinent data about salience and diagnosticity.

```
hasAJQuale(?x,qij) ->  hasQIJSal(?x,sij)

hasAJQuale(?x,qij) ->  hasAJDiag(?x,dj)
```

In Hampton's view, we may not have discrete information about `hasAJQuale(?x,?y)`, we rely on `hasQJDeg(?x,?y)` as inputs of the mechanism.

By means of SWRL constraints, as we have seen in Section 4, we can compute the proximity to the prototype (according to Tversky or Hampton), using the relevant data that are stored by the instantiated data properties.

To integrate SWRL computations within the OWL reasoner – and in doing so, enable reasoning by using the information provided by the background ontology – we add to the ontology the class C, corresponding to the set of entities that are classified under C by means of its prototype. To obtain the extension of the concept, we have to set a threshold t. We may do that by adding the following SWRL constraints:

```
hasHVal(?x,?y)
^swrlb:greaterThanOrEqual(?y,t) -> C(?x)     (H)

hasTVal(?x,?y)
^swrlb:greaterThanOrEqual(?y,t) -> C(?x)     (T)
```

By means of those constraints, the class C of the ontology can be populated with all the instances that meet the threshold, according to the similarity equations, either (2) or (5), and to the equivalence (6). The class C is in general placed as a subclass of POB in DOLCE taxonomy. The reason is that C contains entities that have physical qualities, which must be ascribed to physical objects according to DOLCE. However, if we know that we are classifying more specific types of entities, a more refined placement within DOLCE hierarchy can be decided.

E.g., if we know that we are classifying animals, C can be placed as a subclass of AGENTIVEPHYSICALOBJECT, which refines POB and enables more inferences, cf. [18]. This information is independent of the prototype mechanism and allows for feeding background information into prototypes.

The previous rules (H) and (T) establish "sufficient" conditions for membership to C, i.e. if an instance reaches a certain C-value, then it is a C. Setting a necessary condition for membership to C is not straightforward. We

[7] We have introduced the object properties `hasAJQuale`, instead of a single `hasQuale` relation, to facilitate the statement that an object has exactly one quale of each type. In OWL, we could also define `hasQuale` in terms of `ql` and `qt` by means of property chains, to be closer to the treatment DOLCE. The use of `hasAJQuale` simplifies the presentation.

cannot write a rule that states that "if C(?x), then ?x must have a value ?y that is greater than the threshold t" because, in SWRL, all the variables in the consequent must appear in the antecedent. We leave this point for future work. However, here we content ourselves with sufficient conditions, also to enable the ontology to infer that something is a C, without explicitly knowing its H- or T-values.

Once the class C is populated by means of the SWRL rules, then we can treat it as a usual class of the ontology, and use any reasoner to obtain the properties of C-entities that are entailed wrt. the background ontology.

Note that, to execute any SWRL rules, each variable occurring in the antecedent has to be instantiated. Since hasQJDeg range in $[-1, 1]$, according to Hampton, instances that are deemed not have the j-th attribute value, might be set to -1. In the case of binary (yes/no) ascription of attribute values, hasQJDeg ranges in $\{0, 1\}$. Stating that an entity does not have feature j amounts to setting the value of hasQJDeg to 0, which entails not weighing the w_j in the computation of proximity.

5.1. Semantic observations

We have proposed a theory based on OWL and SWRL to enable reasoning about the attributes and attribute-values occurring in prototype theories. As usual in applied ontology, ontological theories are used to specify meanings of concepts, mainly by excluding unintended models, see [23]. For this reasons, it is interesting to have a look at the models of the theory.

An (abstract) interpretation of OWL, cf. [14], is given by:

$$\mathcal{I} = (R, EC, ER, L, S, LV)$$

where R is a non-empty set of resources (the domain of $\mathcal{I}$), $LV \subseteq R$ is a set of typed literals (for datatypes), EC is a mapping from classes and datatypes to R and LV (respectively), ER is a mapping from (object and data) properties to relations on R, L is a mapping from typed literals to elements of LV, and S is a mapping from individual names to $EC(\text{owl: Thing}) \subseteq R$.

We refer to [24] and [25] for the usual semantics of OWL. An interpretation satisfies an ontology iff it satisfies all axioms and facts in it. We rephrase how to extend the interpretation to SWRL constraints, cf. [14].

To interpret SWRL atoms, the interpretation $\mathcal{I}$ is extended to $\mathcal{I}'$, assigning an interpretation also to i-variables and d-variables, as follows: *i)* $S(x) \in EC(\text{owl: Thing})$, for any i-variable x; *ii)* $L(y) \in LV$, for any d-variable y. The conditions of satisfaction of SWRL atoms, given $\mathcal{I}'$, are the following ones, for concept description C, object property R, and data property Q:

$C(x)$	$S(x) \in EC(C)$
$R(x, y)$	$(S(x), S(y)) \in ER(R)$
$Q(x, y)$	$(S(x), L(y)) \in ER(Q)$

$\mathcal{I}'$ satisfies an antecedent a1, ..., an of a rule iff it is empty ($n = 0$) or $\mathcal{I}'$ satisfies every atom ai. $\mathcal{I}'$ satisfies a consequent b1, ..., bn iff it is empty or $\mathcal{I}'$ satisfies every atom bi.

A rule is satisfied by an interpretation $\mathcal{I}$ iff, for every extension $\mathcal{I}'$ of $\mathcal{I}$, if $\mathcal{I}'$ satisfies the antecedent, then $\mathcal{I}'$ satisfies the consequent.

An interpretation satisfies an ontology containing (SWRL) rules iff it satisfies every axioms and facts of the ontology and every rules.

Let C be a concept name populated by means of (e.g.) Hampton's similarity and by means of the thresholding constraint, cf. rule (H) with threshold t, where $t \in LV$. Given an interpretation $\mathcal{I}$, the extension of C in $\mathcal{I}$ can be computed by means of $EC_{\mathcal{I}'}$ and it shall include the set of entities that satisfy the rule (H), for every extension $\mathcal{I}'$.

$$EC_{\mathcal{I}'}(C) \supseteq \{r \in EC_{\mathcal{I}'}(\text{owl: Thing}) \mid \text{there exists } l \in LV, (r, l) \in ER_{\mathcal{I}'}(\texttt{hasHVal}) \text{ and } (l, t) \in ER_{\mathcal{I}'}(\texttt{swrlb:greaterThanOrEqual})\}$$

This definition is standard for SWRL and the interpretation of the concept C is termed *knowledge-independent* in [22], as the extension of C simply depends on $\mathcal{I}'$ and not on the knowledge expressed by the background ontology.

Our interpretations of interest are those that satisfy the background ontology DOLCE as well as the instantiated data properties. We denote by D the ontology obtained by joining DOLCE with the required facts about the instantiated data properties. Let $\mathcal{I}_D$ any interpretation that satisfies D. The rules that we have introduced are intended to be interpreted wrt. interpretations that satisfy D, i.e $\mathcal{I}'_D$. However, deciding a single particular interpretation is too demanding. For instance, $\mathcal{I}'_D$ may associate values of $ER_{\mathcal{I}'}(\texttt{hasHVal})$ for non-named individuals.

What we wish is to allow for *any* interpretation that satisfies D and that applies the prototype classifications explicitly to those individuals for which we have actually instantiated the data properties.

This view of the interpretation of C is termed *knowledge-dependent* in [22]. We can define the set of individual names that are included in C, wrt. any interpretation that satisfies D, by leveraging on the logical consequences ($\models$) of D:[8]

$$D \models Ca \text{ if exists l s. t. } D \models \texttt{hasHVal(a,l)} \text{ and } D \models \texttt{swrl:greaterThanOrEqual(l,t)}\}$$

[8] Cf. Definition 4 in [22].

In this case, `l` and `t` are individual names, which are interpreted as data literals. There is a more standard way to present the definition of a knowledge-dependent interpretation of C in terms of interpretations, cf. [22], Definition 5. We leave the detailed discussion of this point to future work.

6. Example: "Elephant"

We render the prototype-based concept `Elephant` in OWL plus SWRL, discussing, for reasons of space, only the variants proposed by Hampton and Rosch, while we leave Tversky's rule for future applications, cf. Section 2.[9]

In this case, a prototype is a list of features associated with weights, expressing their importance. Simplifying, suppose that the prototype description of `Elephant` is:

$$\texttt{Elephant} = \{(\texttt{hasTrunk}, 3), (\texttt{Large}, 2), (\texttt{Grey}, 2)\}$$

The description mentions two features that we can interpret as two attributes (`Size` and `Color`) with attribute-values (`large` and `grey`). Our background ontology includes the taxonomy of DOLCE and it partitions the class `PhysicalRegion` into two subclasses `Size` and `Color`, containing as elements (individual names) `large` and `grey` (respectively). We also have a feature `hasTrunk`, which can be interpreted ontologically in terms of parthood, rather than as a quality structure. The feature `hasTrunk` is interpreted by the defined class $\exists$`hasPart.Trunk`, where `Trunk` is a subclass of POB and `hasPart` is the mereological relation of DOLCE. Moreover, we introduce the class `Elephant` and we may assume that it is placed under the class `AgentivePhysicalObject` of DOLCE, cf. [18], to leverage the inferences that are enabled by this classification.

To render prototypical classifications of `Elephant` by means of SWRL, we introduce the following (functional) data properties:

`hasTrunkWei`, `hasLargeWei`, and `hasGreyWei`
`hasTrunkDeg`, `hasLargeDeg`, and `hasGreyDeg`
`hasTrunkVal`, `hasLargeVal`, and `hasGreyVal`

The values of entities on each feature are then computed by means of the following three SWRL constraints:

```
hasTrunkDeg(?x,?d) ^ hasTrunkWei(?x,?w)
^ swrlb:multiply(?v,?d,?w)
->  hasTrunkVal(?x,?v)

hasLargeDeg(?x,?d) ^ hasLargeWei(?x,?w)
^ swrlb:multiply(?v,?d,?w)
->  hasLargeVal(?x,?v)

hasGreyDeg(?x,?d) ^ hasGreyWei(?x,?w)
^ swrlb:multiply(?v,?d,?w)
->  hasGreyVal(?x,?v)
```

Notice that this data allow us to represent both Hampton's and Rosch's similarity measures. The difference in the two measures is in the range of the data properties establishing degrees: while in the case of Hampton, the value ranges in the interval $[-1, +1]$, in the case of Rosch it is in {0,1} (1, if the instance has the feature, 0 if it doesn't).

To link data properties with object properties, we may add the following rules:

```
hasGreyDeg(?x,1) -> hasColorQuale(?x,grey)
hasLargeDeg(?x,1) -> hasSizeQuale(?x,large)
hasTrunkDeg(?x,1) -> hasPartSomeTrunk(?x)
```

Where `hasPartSomeTrunk` is a defined class, characterised by $\exists$`hasPart.Trunk`.[10]

Then, the data property `hasElephantVal` is introduced to compute the Hampton similarity of instances with the prototype.

The values associated with this data property are computed by the SWRL constraint:

```
hasTrunkVal(?x,?v1) ^ hasLargeVal(?x,?v2)
^ hasGreyVal(?x,?v3) ^
swrlb:add(?v,?v1,?v2,?v3)
-> hasElephantVal(?x,?v)
```

Suppose we are given a threshold `t` for the membership to the class defined by the prototype. We can then populate the class `Elephant` by means of the following SWRL constraint:

```
swrlb:greaterThanOrEqual(?y,t)
^ hasValueElephant(?x,?y) -> Elephant(?x)
```

Let us suppose we have an instance (an individual name) for Dumbo, "`dumbo`". We might get different descriptions of dumbo depending on the Prototype theory version we are considering. In the case of Rosch, we might have that: `hasTrunkDeg(dumbo,1)`, `hasLargeDeg(dumbo,0)` and `hasGreyDeg(dumbo,1)`.

In this case, we get:

```
hasTrunkValue(dumbo,3),
hasLargeValue(dumbo,0),
hasGreyValue(dumbo,2)
```

[9] The ontology implementing this toy example can be found at https://github.com/diporello/OntologyForPrototypesSWRL/blob/main/TestPrototypeTheoryOntologySWRL.owx. The documentation for the SWRL plug-in can be found at https://github.com/protegeproject/swrltab-plugin.

[10] This definition is required to run the SWRL plug-in for Protégé.

Regarding Hampton's similarity measure, we might have, e.g.:

```
hasTrunkDeg(dumbo,1),
hasLargeDeg(dumbo,0.3),
hasGreyDeg(dumbo,0.9).
```

In this case, we obtain:

```
hasTrunkVal(dumbo,3),
hasLargeVal(dumbo,0.6),
hasGreyVal(dumbo,1.8)
```

If the threshold t is set to 4 ($L(t) = 4$), then the SWRL engine can infer that the threshold is met and that Dumbo is thus included (in both cases) as an instance of the class `Elephant`. That is, the class `Elephant` is populated by the instance `dumbo`, which then inherits the ontological description coming from the axioms of the background ontology. For crisp ascriptions of degrees, the ontology will also populate the following object properties:

```
hasTrunkDeg(dumbo,1) -> hasPartSomeTrunk(dumbo)

hasGreyDeg(dumbo,1) -> hasColorQuale(dumbo,grey)
```

7. Discussion: Learning Weights

In the above examples, it was assumed that the weights associated with the prototype concepts (modelled according to the different variants) were given. We depict a scenario where we can learn a prototype concept from data. If the prototype concept to be learned is treated as a linear classification model, then it is possible to use standard linear classification algorithms (such as the perceptron algorithm, logistic regression, or linear SVM) to learn its weights and its threshold, given a set of assertions or facts about individuals.

This setting was introduced in [22], where the procedure for learning a prototype concept of an 'Elephant' was described as follows. Assume we have a knowledge base $\mathcal{K}$, a target concept P_T, and a finite number of concept features $C_1, \ldots, C_n$ (e.g., has a trunk, has a tusk, is of size large, is of color grey, etc.) representing the various concepts that may appear in a prototypical definition of 'Elephant', which is denoted by P_T. Concepts C_i may be defined, that is they may represent attribute-values ascriptions such as `hasAJQuale(?x,q)` or more general features such as `hasPartSomeTrunk(?x)`.

Then, each individual a in the knowledge base $\mathcal{K}$ induces a pair $\langle \vec{x}(a), y(a) \rangle$ where $\vec{x}(a) \in \{0,1\}^n$ is such that its i-th element $\vec{x}(a)_i = 1$ if $\mathcal{K} \models C_i(a)$, and it is 0 otherwise; likewise, $y(a) = 1$ if $\mathcal{K} \models P_T(a)$, and $y(a) = 0$ otherwise. This setting applies to the case where classifications under features or attribute-values are binary.

Based on this encoding, a data set can be defined containing positive and negative examples about the concepts that may appear or not as the concepts *required* by the description of an 'Elephant'[11].

Then, the task will be to learn a linear classification model of the sort $y(a) = 1$, if $\sum_i x(a)_i w_i + c \geq 0$, $y(a) = 0$, otherwise, that will approximate P_T on the basis of $C_1, \ldots, C_n$. Once the weights $w_1, \ldots, w_n$ and the constant term c are learned, these can be used to define the prototype concept as $\{(C_1, w_1), \ldots, (C_n, w_n)\}$ where $t = -c$ is set as the threshold for the membership to the class defined by P_T.

For the case of the Tversky measure, that requires to differentiate the contribution of salience and diagnosticity, we could assume that the salience of a feature is 'learned' from domain experts, e.g. by asking them what attribute-values make the difference in classifying a certain concept, or obtained from the numbers of instances that are present in the data. The diagnosticity could be treated, instead, as the weight of a feature. In this way the previous learning scenario could be applied.

A different setting was illustrated in [27] where the authors proposed to learn Tversky's similarity (and a weighted variant of it) of two images based on the concept features contained in the images. In their setting, the learning data are represented by pairs of sets of features which inform about the semantic similarity or dissimilarity between objects. Then, the problem of learning the semantic similarity between two objects is modelled as an optimization problem using a contrastive loss function. This approach could be generalised to the case in which we learn a similarity metric from a given set of instances of a prototype. The similarity measure could then be used as a threshold classifier to decide for the membership of an instance to the class defined by a given prototype concept P_T.

A recent related work is dedicated to study the compilation problem of linear classifiers into Ordered Binary Decision Diagram (OBDD), [28]. Training of single neurons with integers weights and thresholds has also been studied from the motivation of explainability [28], namely looking at the knowledge compilation problem to determine under which conditions a neuron can be efficiently trained and then compactly represented as OBDD. A comparison between the learnability of OBDD and threshold concepts is an interesting line of future work.

[11] The paper [26] proposes a method for learning the weights in a way that such requirements can be captured.

8. Conclusion and future work

We presented an environment based on OWL 2 and SWRL to implement the main aspects of prototype theories. We used data properties to store information about weights, degrees, and salience, and we used rules and the built-in SWRL operators to compute the similarity between instances. As we have demonstrated, the SWRL setting is sufficient to implement the main aspects of prototype theories. SWRL constraints allow also for populating the ontology (its classes and object properties) with information that is deemed sufficiently crisp. By adopting a background ontology, we enabled a certain level of formal description of the attributes, the attribute-values, and the general features occurring in the prototype descriptions. Also, we introduced the means to integrate prototypical classifications with ontological, feature-independent information.

Future work is planned in three directions. Firstly, we are interested in exploring the learning algorithms for weights, salience, and diagnosticity values. Secondly, we plan to approach threshold and typicality operators, cf. [7, 11, 8, 6], to implement them, or approximate them, by means of SWRL, and to extend our approach to more expressive threshold operators, e.g. including counting capabilities [29]. Finally, we are interested in exploring the SWRL capacity to provide an ontological rendering of the exemplar view of concepts, cf. [30].

Acknowledgments

This research is partially supported by Italian National Research Project PRIN2020 2020SSKZ7R: BRIO – Bias, Risk and Opacity in AI, https://sites.unimi.it/brio/

References

[1] G. Murphy, The big book of concepts, MIT press, 2004.

[2] E. Rosch, C. B. Mervis, Family resemblances: Studies in the internal structure of categories, Cognitive psychology 7 (1975) 573–605.

[3] E. E. Smith, D. N. Osherson, L. J. Rips, M. Keane, Combining prototypes: A selective modification model, Cognitive science 12 (1988) 485–527.

[4] J. A. Hampton, Testing the prototype theory of concepts, Journal of Memory and Language 34 (1995) 686–708.

[5] C. Masolo, D. Porello, Representing concepts by weighted formulas, in: Formal Ontology in Information Systems, IOS Press, 2018, pp. 55–68.

[6] D. Porello, O. Kutz, G. Righetti, N. Troquard, P. Galliani, C. Masolo, A toothful of concepts: Towards a theory of weighted concept combination, in: M. Simkus, G. E. Weddell (Eds.), Proceedings of the 32nd International Workshop on Description Logics, Oslo, Norway, June 18-21, 2019, volume 2373 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2019.

[7] F. Baader, A. Ecke, Reasoning with Prototypes in the Description Logic $\mathcal{ALC}$ Using Weighted Tree Automata, in: Language and Automata Theory and Applications: 10th International Conference, LATA 2016, Prague, Czech Republic, March 14-18, 2016, Proceedings 10, Springer, 2016, pp. 63–75.

[8] A. Lieto, G. L. Pozzato, A description logic framework for commonsense conceptual combination integrating typicality, probabilities and cognitive heuristics, Journal of Experimental & Theoretical Artificial Intelligence 32 (2020) 769–804.

[9] L. Wittgenstein, Philosophical Investigations, Oxford: Blackwell, 1953.

[10] K. Britz, J. Heidema, T. Meyer, Modelling object typicality in description logics, in: AI 2009: Advances in Artificial Intelligence: 22nd Australasian Joint Conference, Melbourne, Australia, December 1-4, 2009. Proceedings 22, Springer, 2009, pp. 506–516.

[11] L. Giordano, V. Gliozzi, N. Olivetti, G. L. Pozzato, A non-monotonic description logic for reasoning about typicality, Artificial Intelligence 195 (2013) 165–202.

[12] C. A. Yeung, H. Leung, Ontology with likeliness and typicality of objects in concepts, in: D. W. Embley, A. Olivé, S. Ram (Eds.), Conceptual Modeling - ER 2006, 25th International Conference on Conceptual Modeling, Tucson, AZ, USA, November 6-9, 2006, Proceedings, volume 4215 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 98–111. URL: https://doi.org/10.1007/11901181_9. doi:10.1007/11901181_9.

[13] Y. Cai, H. Leung, A. W. Fu, Multi-prototype concept and object typicality in ontology, in: D. Wilson, H. C. Lane (Eds.), Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society Conference, May 15-17, 2008, Coconut Grove, Florida, USA, AAAI Press, 2008, pp. 470–475. URL: http://www.aaai.org/Library/FLAIRS/2008/flairs08-111.php.

[14] I. Horrocks, P. F. Patel-Schneider, S. Bechhofer, D. Tsarkov, OWL rules: A proposal and prototype implementation, J. Web Semant. 3 (2005) 23–40. URL: https://doi.org/10.1016/j.websem.2005.05.003. doi:10.1016/j.websem.2005.05.003.

[15] M. J. O'Connor, R. D. Shankar, M. A. Musen, A. K. Das, C. Nyulas, The SWRLAPI: A development environment for working with SWRL rules, in: C. Dolbear, A. Ruttenberg, U. Sattler (Eds.), Proceedings of the Fifth OWLED Workshop on OWL: Experiences and Directions, collocated with the 7th International Semantic Web Conference (ISWC-2008), Karlsruhe, Germany, October 26-27, 2008, volume 432 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2008. URL: https://ceur-ws.org/Vol-432/owled2008eu_submission_41.pdf.

[16] M. J. O'Connor, H. Knublauch, S. W. Tu, B. N. Grosof, M. Dean, W. E. Grosso, M. A. Musen, Supporting rule system interoperability on the semantic web with SWRL, in: Y. Gil, E. Motta, V. R. Benjamins, M. A. Musen (Eds.), The Semantic Web - ISWC 2005, 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005, Proceedings, volume 3729 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 974–986. URL: https://doi.org/10.1007/11574620_69. doi:10.1007/11574620_69.

[17] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, WonderWeb Deliverable D18, Technical Report, CNR, 2003.

[18] S. Borgo, R. Ferrario, A. Gangemi, N. Guarino, C. Masolo, D. Porello, E. M. Sanfilippo, L. Vieu, DOLCE: A descriptive ontology for linguistic and cognitive engineering, Applied ontology 17 (2022) 45–69.

[19] D. N. Osherson, E. E. Smith, On the adequacy of prototype theory as a theory of concepts, Cognition 9 (1981) 35–58.

[20] P. Gärdenfors, Conceptual spaces - The geometry of thought, MIT Press, 2000.

[21] G. Righetti, O. Kutz, D. Porello, N. Troquard, A Game of Essence and Serendipity: Superb Owls vs. Cooking-Woodpeckers, in: M. M. Hedblom, A. A. Kantosalo, R. Confalonieri, O. Kutz, T. Veale (Eds.), Proceedings of the 13th International Conference on Computational Creativity, Bozen-Bolzano, Italy, June 27 - July 1, 2022, Association for Computational Creativity (ACC), 2022, pp. 300–309. URL: http://computationalcreativity.net/iccc22/papers/ICCC-2022_paper_88.pdf.

[22] P. Galliani, O. Kutz, D. Porello, G. Righetti, N. Troquard, On knowledge dependence in weighted description logic, in: D. Calvanese, L. Iocchi (Eds.), GCAI 2019. Proceedings of the 5th Global Conference on Artificial Intelligence, Bozen/Bolzano, Italy, 17-19 September 2019, volume 65 of *EPiC Series in Computing*, EasyChair, 2019, pp. 68–80. URL: https://doi.org/10.29007/hjt1. doi:10.29007/hjt1.

[23] N. Guarino, M. Carrara, P. Giaretta, Formalizing ontological commitment, in: B. Hayes-Roth, R. E. Korf (Eds.), Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 1, AAAI Press / The MIT Press, 1994, pp. 560–567. URL: http://www.aaai.org/Library/AAAI/1994/aaai94-085.php.

[24] F. Baader, I. Horrocks, C. Lutz, U. Sattler, Introduction to description logic, Cambridge University Press, 2017.

[25] P. Patel-Schneider, OWL web ontology language semantics and abstract syntax W3C recommendation 10 february 2004, http://www. w3. org/TR/2004/REC-owl-semantics-20040210/ (2007).

[26] G. Righetti, P. Galliani, C. Masolo, Concept Combination in Weighted DL, in: S. A. Gaggl, M. V. Martinez, M. Ortiz (Eds.), Logics in Artificial Intelligence - 18th European Conference, JELIA 2023, Dresden, September 20-22, 2023, Proceedings, volume 14281 of *Lecture Notes in Computer Science*, Springer, 2023.

[27] J. Rahnama, E. Hüllermeier, Learning tversky similarity, in: M.-J. Lesot, S. Vieira, M. Z. Reformat, J. P. Carvalho, A. Wilbik, B. Bouchon-Meunier, R. R. Yager (Eds.), Information Processing and Management of Uncertainty in Knowledge-Based Systems, Springer International Publishing, Cham, 2020, pp. 269–280.

[28] L. Kennedy, I. Kindo, A. Choi, On Training Neurons with Bounded Compilations, in: Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning (KR 2023), 2023, pp. 395–405. URL: https://doi.org/10.24963/kr.2023/39. doi:10.24963/kr.2023/39.

[29] P. Galliani, O. Kutz, N. Troquard, Succinctness and Complexity of $\mathcal{ALC}$ with Counting Perceptrons, in: Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning (KR 2023), Rhodes, Greece, September 2–8, 2023. URL: https://doi.org/10.24963/kr.2023/29. doi:10.24963/kr.2023/29.

[30] D. L. Medin, M. M. Schaffer, Context theory of classification learning., Psychological review 85 (1978) 207.