# The unique Predication of Knowledge Elements and their Visualization and Factorization in Ontology Engineering

Hermann Bense
*bense.com digital publishing GmbH, Dortmund, GERMANY*
*E-Mail: hb@bense.com, Web: www.ontology4.us*

**Abstract**. In knowledge representation nowadays there exists no common notation which is used to attach designations and meanings to knowledge elements. E.g. the name *painting* can stand either for the class *painting* or for the object property (relation) *painting* which assigns *paintings* to their *painters*. For a clear distinction typically syntactical overhead is needed. In OWL/XML-Syntax this would be <owl:Class rdf:id="Painting"> </owl:Class>. Therefore a set of special characters is introduced, which in combination with a name allows to uniquely designate different types of knowledge elements with the same name like ^Painting (class) or <>Painting (relation). The character combination >> is used to designate n-ary relationships like >>Nikolaus_Kopernikus-Belief. Using this designation conventions, we propose the notation of an Ontological Graph (OG), which is powerful enough to visualize data models, semantic networks, conceptual roles, formulas, process models, situations, taxonomies etc. Based on the type of knowledge elements, nodes have specific colors and shapes and can be enriched by graphical elements, images and hypertext links. OGs allow for a kind of visual thinking and support to develop, debug, document and visualize knowledge faster, easier and more consistently. Finally we present an approach for a predication system (LPS), which allows to attach unique predicators to knowledge elements based on Leibniz characteristic numbers (LCN). This allows to construct, designate and retrieve knowledge elements from primitive ones by means of factorization e.g. *creature = living#body*, which means, that we can support search by meaning.

**Keywords.** Knowledge Element, Knowledge Representation, Naming Convention, Ontology Development and Visualization, Semantic Web, Semiotic Triangle

## 1. Introduction

All examples of this article have been edited and visualized with the Knowledge Based Content Management System (KB-CMS) Ontology4 [1]. Ontology4 is a knowledge engineering environment for the development of web applications based on ontologies according to the ideas of the semantic web. The kernel of the system is a content management system with a transparent access to the underlying knowledge repository (KR). The information of the knowledge repository (KR) can be used as a source for dynamically integrating data into documents by means of so called OntoPages as well as doing inferences, proofs and calculations. Within Ontology4 for the ease of use and non ambiguous uniqueness we propose naming conventions for the most

important knowledge elements. For each fundamental knowledge element like class, attribute, relationship, process, rule, set, situation etc. special characters are reserved to associate meaning to names of knowledge elements [Table 1]. The underlying knowledge for the generation of the graphs is maintained in a quadruple store repository. There is only one table in every Ontology4 knowledge base with the following structure: **SROP (Subject, Relation, Object, Period)** where Period = (Date, Time, ToDate, ToTime). In contrast to the implementation of systems using RDF, which are based on triple stores, we chose to add the date and time attributes (time period or point in time) for the following reasons: Very often relationships between subjects and objects are valid for a specific time period or represent events like .DateOfBirth, <>married, <>believes etc. In these cases we don´t have a need to represent the date and time attribute in additional relationships. This allows to make decisions on the base of the information, whether a relation has a beginning or an end date or time. We will also show some examples of how to model nested conceptual relations (CR) like beliefs and wishes.

| Type | Description | Symbol |
|---|---|---|
| class_ | **Class / Concept**: A knowledge element name prefixed by a ^ character represents a class, etc. Classes are always colored dark red. The motivation for the circumflex is that it points upwards to a higher concept. |  |
| att_ | **Attributes / Properties**: knowledge element name prefixed by a dot (.) represents an attribute and is display as a green octagon. The motivation for the full stop is that most programming language use it to access attribute values. |  |
| sub_ | **Subject / Object**: A knowledge element name prefixed by a > represents an instance of a class. Instances of classes are colored green. The motivation for the greater-than sign is, that points to a thing, which can be subject or object in a relation. |  |
| set_ | **Set**: The motivation for the >? is, that it represents a set of instances. |  |
| rel_ | **Relationships**: The motivation for the <> character combination is that it represents a bi-directional relationship. |  |
| sit_ | **Situation**: A knowledge element name prefixed by the special character combination >> represents a situation or theme. It is used to connect other knowledge elements in n-ary relations. |  |
| act_ | **Activities and Processes**: A knowledge element name prefixed by a circle (°) represents a process or activity and is displayed as an orange house. The motivation for the degree sign is that its circle shape represents things, which can last forever. |  |
| impl_ | **Implementation**: A knowledge element name prefixed with the character combination (>°) represents the implementation (algorithm) aspect of a method, rule, a condition or a formula. |  |

**Table 1.** Special symbols for designation of knowledge elements

## 2. Using Acronyms for the Naming of Knowledge Elements

There are many different usages of terms in ontology engineering. Instead of using terms like ***concept*** we prefer to use ***knowledge elements*** [2]. At first we introduce the idea of naming knowledge elements with acronyms and special characters. For the purpose of a more precise naming of knowledge elements in ontologies we have collected examples of acronyms [3]. The table contains two to five letter abbreviations for knowledge element types which appear in considerable numbers in knowledge worlds such like kinds of animals (ANM), airports (ARP), books (BOK), buildings (BLD), branches (BRN), cities (CIT), countries (CNT), persons (NPS = natural person), professions (PRO), publications (PUB), rivers (RVS), streets (STR), universities (UNV), webpage addresses (URL) and many more. In the following we introduce a set of naming conventions. Instead of using the too general term ***name*** we will either use ***designator*** or ***predicator*** [4]. We avoid the term ***identifier*** since it is too much pre-allocated with its use in computer science.

Furthermore, we use the term ***instance*** for a knowledge element which has been instantiated from a class. We would like to emphasize that here the use of the term ***instance*** might differ from the use in object oriented programming. Designators of classes start with the special character **^** and designators of instances start with the special character **>**. Examples of class designators are ^NPS (natural person), ^Artist, ^Painter, ^Work, ^ArtWork, ^Painting, ^Building, ^Museum etc. When a designator like ^Painter appears, one automatically knows that it designates a class. No additional syntax or meta information is needed. The same holds for instances. **>NPS-Pablo_Picasso** is the designator for the natural person ***Pablo Picasso***. In case that there are several Pablo Picassos, one could simply enumerate like >NPS-Pablo_Picasso_1, >NPS-Pablo_Picasso_2, >NPS-Pablo_Picasso_3 etc. Thus simply by its designator one knows that the instance referred to by >**NPS**-Pablo_Picasso is a ***natural person*** identified by the acronym **NPS**. This is also modelled by the knowledge atom **(>NPS-Pablo_Picasso, <>isi, ^NPS)**. The relation **<>isi** is an abbreviation for **<>is_instance_of**. The same principle for designating instances can also be applied for the designation of classes in large class hierarchies. Branches (BRN), professions (PRF), animals (ANM) etc. have many subclasses and can form big taxonomies. Therefore it is appropiate to designate classes with designators like ^ANM_Bird, ^ANM_Fish, ^ANM_Insect, ^ANM_Mammal, ^ANM etc. Then by its designator acronym it can be automatically induced, that a fish (^ANM_Fish) is an animal. In the knowledge base this can be explicitly documented by (^ANM_Fish, <>Acronym, >ACR_ANM) where (>ACR_ANM, .Name, **animal**) establishes the reference to the word ***animal***.

## 3. The "Pablo Picasso" small world ontology example

The example in Fig. 1 demonstrates how we use the designation conventions in a real world excerpt about Pablo Picasso and his work. The relation **>NPS-Pablo_Picasso, <>isi, ^Painter)** represents the fact, that the subject Pablo Picasso (>NPS-Pablo_Picasso) is an instance (relation name: **<>isi)** of the object class painter (**^Painter**). Instances like **>NPS-Pablo_Picasso** and **>ATW_Guernica** are presented as rounded rectangles with a green title bar. Relationships between instances are represented by blue directed arcs, e.g. **<>is_Painting_of**. They are the result of an activity like **~paint** which is a method of the class **^Painter**. The relation **<>isi** connects

an instance to its class with a green arc. A class is represented as a dark red ellipse. Attribute names of instances are written in green. The special attribute **.Image** allows to integrate pictures of common file formats like .jpg, .gif, .png etc. Instances can have attributes (data properties), where the prefix character is a dot (**.**) and relations to other instances where the prefix characters are <>.

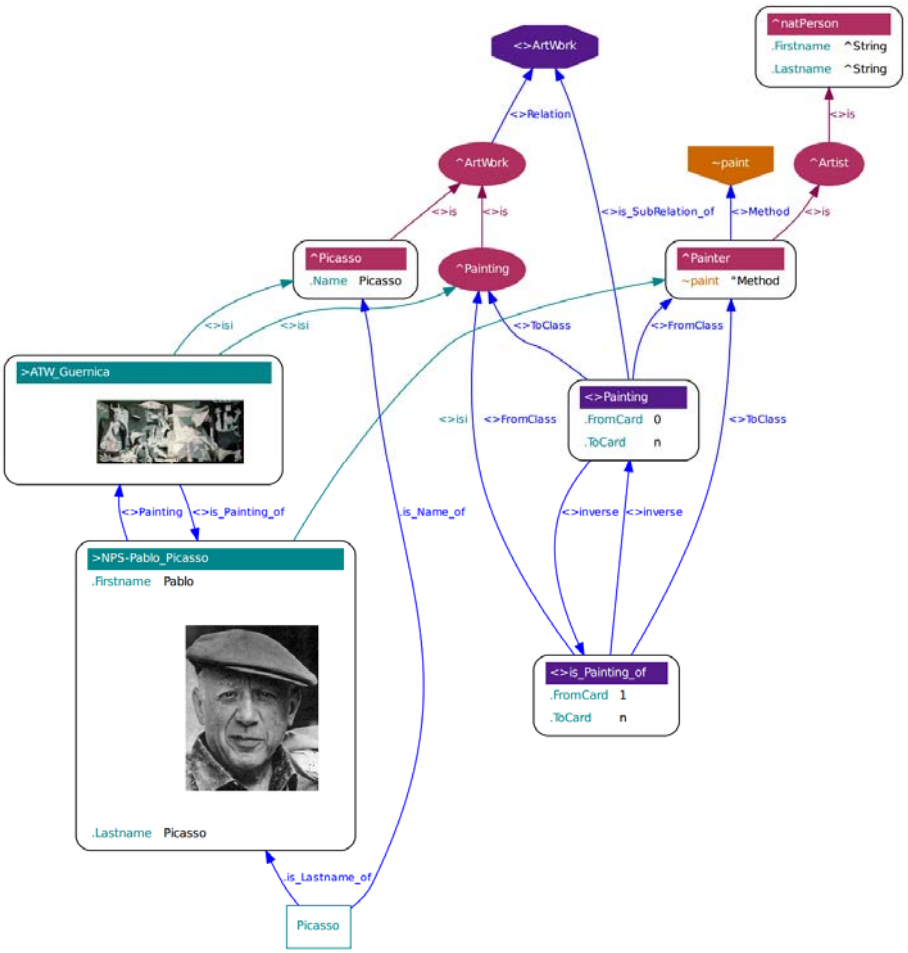



Fig. 1 The "Pablo Picasso" small world ontology example

Relations (object properties) between classes specify, what is the origin (**<>FromClass**) and what is the target (**<>ToClass**) of a relationship. E.g. the relation **<>Painting** connects **^Painter** and **^Painting** and has .FromCard=0 and .ToCard=n, which means, that a painter can have painted 0 to n paintings. The **<>inverse** relation **<>is_Painting_of** represents the opposite direction. A painting can have been painted by 1 to n painters. That means, that a painting cannot exist without a painter. As a speciality in naming conventions one could also define a class **^Picasso**. This would stand for the artworks, which Pablo Picasso created, e.g. **(>ATW-Guernica, <>isi, ^Picasso)** which means *Guernica is a Picasso*. In this case the word *Picasso* is part of the predicator

**^Picasso** and also the value of the attribute .Lastname of the instance >NPS-Pablo_Picasso. The three nodes labeled **^Picasso**, **Picasso** and **>ATW_Guernica** are a typical example for a semiotic triangle, with **Picasso** being the symbol (literal representation), **^Picasso** being the thought of reference and **>ATW_Guernica** being the referent. The OG (Ontological Graph) of Fig. 1 has been generated with the GraphViz library [5] on the base of the contents of the knowledge base. The OG in Fig. 2 displays the colors and forms for knowledge elements, which we discuss in this paper. It also visualizes the typical relations between the different types of knowledge elements. Compared to the Ontological Sextet [6] as primitive knowledge elements we have added **relations** and **relators** and use different names for basic knowledge elements. But we think that with the set of knowledge elements and relations described in Table 1 and Fig. 2 we are close to a minimal upmost ontology (UMO) [7]. A very important aspect of these kind of OGs is, that knowledge elements can be differentiated by naming conventions, shaping and coloring at the same time. So often it is very easy to find out, if things are properly and consistently modeled. Knowledge elements are connected by different kinds of relations like e.g. <>is, <>SubClass, <>his, <>Attribute etc. Depending on the kind of relations different colors are chosen.

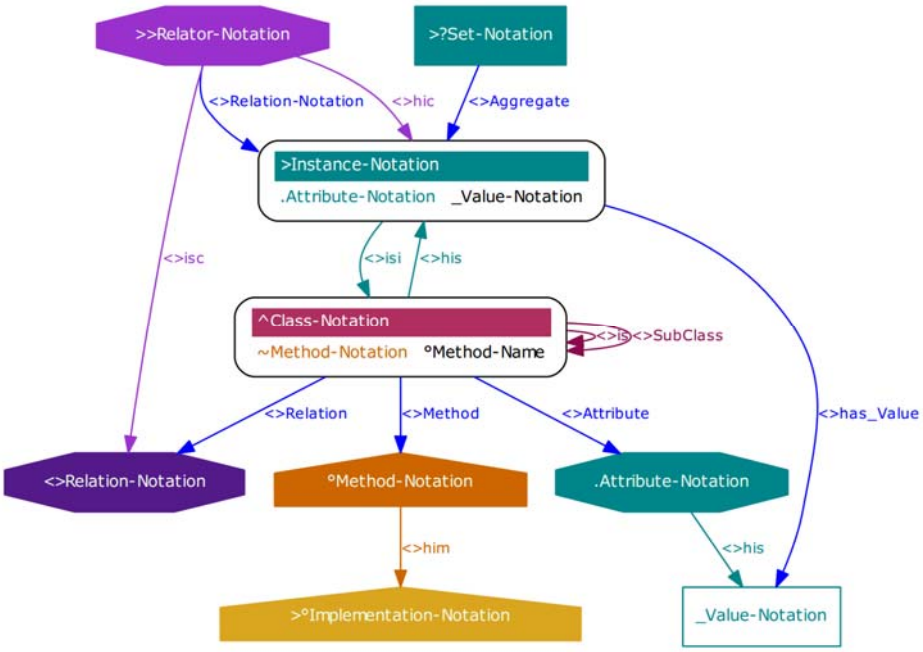


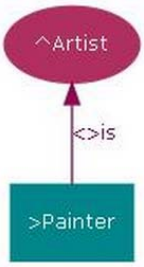

Fig. 2 Types of Knowledge Elements and their representation with forms, colors and special characters

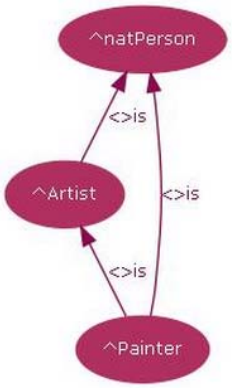
In [8] Svatek et al. present a number of naming suggestions for ontology naming patterns. We will compare them to our approach for naming conventions. The main difference is, that we use unique special characters to classify names and that we differentiate between single properties (data properties) and relationships (object properties): For **case and delimiter conventions** we are quite close to another. Compounds should e.g. use ***camel case*** like in **ComputerMouse** or underscores like in

**Pablo_Picasso**. But for **class naming**, we would use our special character ^ to prefix the class name, e.g. **^Painting**. So we can more easily differentiate, when it comes to relations like **<>Painting**. We preserve the **instance naming** convention to instances of classes and prefix them by >. So in our notation **>NPS-Pablo_Picasso** is clearly an instance of the class **^Painter (>NPS-Pablo_Picasso, <>isi, ^Painter)**. The instantiation pattern therefore always is: **(>instance, <>isi, ^class)** or even shorter **(>, <>, ^)**. For the designation of **data properties** in our approach attributes names are always preceded by a dot (**.**) e.g. in **.Lastname, .Salary, .DateOfBirth** etc. We regard a name like **has_Author** as a relationship name or object property and not as a property name and prefix it with <> e.g. **<>hasAuthor**. In this case a publication is related to a **^natPerson**, which plays the role of an **^Author** in the relation **(^Publication, <>Author, ^natPerson).** For the naming of **object properties** (relations) we heavily recommend to use nouns (and not verb forms like <>works_for) like **<>Author**, **<>Artwork**, **<>Part** etc. whereever possible. Normally these would be abbreviations of the equivalent names **<>has_Author**, **<>has_Artwork**, **<>has_Part** etc. Then it is very easy to find the naming for the inverse relationships by simple applying the pattern **<>inverse(<>*ObjectProperty*)** = **<>is_*ObjectProperty*_of**, e.g. <>is_Author_of, <>is_Artwork_of, <>is_Part_of etc. The disadvantage of using verbs in object property names is, that they always associate a certain tempus like present or past e.g. <>believed, <>created, <>is_married_to etc. The semantics of the relation name can become wrong very fast, if the relationship no longer holds. Using nouns is more neutral and can be combined with the time period, which we can assign to a relation, e.g. **(>Charles_Prince_of_Wales, <>Wife, >Diana_Princess_of_Wales, [1981-07-29|1996-08-28])**

## 4. Network Graph Type Visualizations

In the following we show how the designation conventions can be advantageously used in the visualization of ontologies with Ontological Graphs (OG). The different application areas are data models, process models, rule networks and conceptual graphs (CG), formulas, taxonomies for classes, relationships of persons, semantic networks, social networks etc. Arcs in OGs (Ontological Graphs) are always directed from ***subject*** to ***object*** and are labeled with the name of the ***relation***. This is in contrast to other approaches for the representation of knowledge like e.g. mind maps. It is very helpful to use OGs for the debugging of the knowledge base, since in the generated graphs it can be very easily seen, if there are irregularities in the use of symbols or relations. An example for inconsistency is given in the graph to the left. Obviously it was intended, that **Painter** should be a subclass of **Artist**. Since the wrong character > was chosen to designate the painter class, instead of a red ellipse a green rectangle was chosen for the visualization. The graph to the right shows the redundancy in a small class hierarchy. From the graph representations the problems become obvious very fast. Under [10] we find the RDF primer documentation for the Resource description framework RDF. *RDF is a language for representing information about resources in the World Wide Web* [11, 12]. This



means on the contrary that RDF primarily does not intend to represent knowledge, which does **not** reside in the Web. In the semantic web, resources like the definitions of classes are accessible through there URL. The RDF graph on the left of Fig. 3 is taken from the RDF primer and represents information about the Person Dr. Eric Miller. RDF Graphs use only two different graphical elements, namely green ellipses for things which represent websites or e-mail address and yellow boxes for values. The labels of the relations refer to other website, which describe the object information. In the example of Fig. 3 the predicator for the class **Person** therefore is **^http://www3.org /2000/10/swap/pim/contact#Person**.

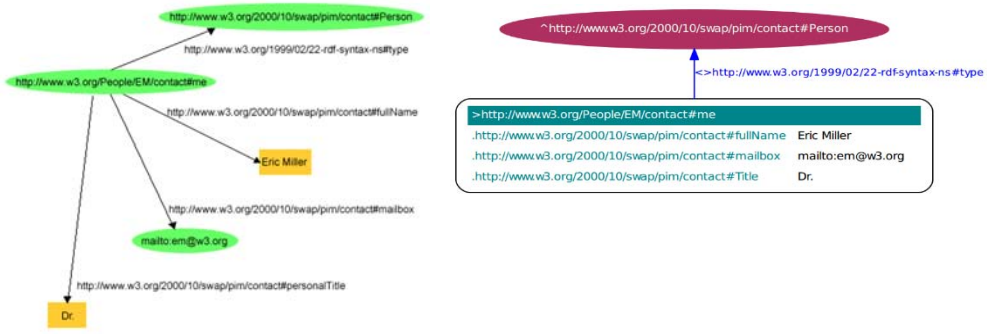



Fig. 3 Comparison of RDF-graphs with Ontology4-graphs (OG)

The graph to the right of Fig. 3 represents the OG for it. It groups the fields #fullName, #mailbox and #Title in one rounded instance rectangle. The type description arc points to the dark red ellipse, where the metadata for **#Person** is contained. Compared to the RDF graph, the OG is more compact while covering the same semantics. In [9] John Sowa compares different graph types for the representation for logics and semantic networks like conceptual graphs (CR), correlational nets, dependency graphs. With Ontological Graphs (OG) we propose an alternative approach for a graphical notation, which is as powerful as CRs but with a more expressive graphic representation.

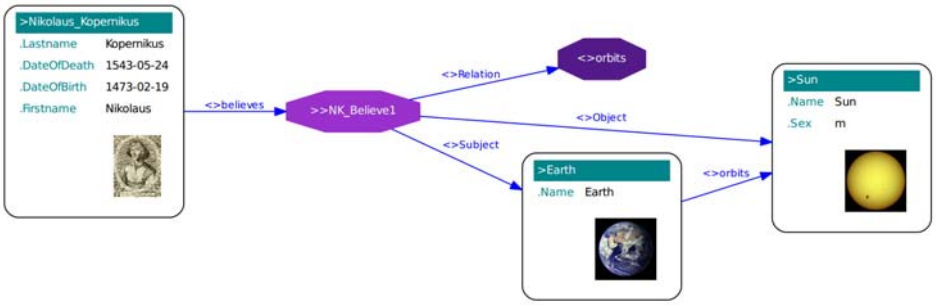



Fig. 4 The n-ary predicator >>NK_Believe1 modeling a belief of Nikolaus Kopernikus

The example in Fig. 4 models the belief of **>Nikolaus_Kopernikus**, that the **>Earth <>orbits** the **>Sun**. The belief is modeled with the n-ary relator and predicator **>>NK_Believe1**, where <>Subject is **>Earth**, <>Object is **>Sun** and the <>Relation between <>Subject and <>Object is **<>orbits**.

## 5. LPS - The Leibniz Predication System

Beyond the problem of unique designation of knowledge elements it is even more challenging to capture the meaning of knowledge elements. We use the terms **designator** and **predicator** as close as possible in the sense of Rudolf Carnap [4]. Kamlah and Lorenzen [13] also did a lot of basic work for the careful usage of names and predicators. Here we use the Leibniz idea for the unique designation of ontological knowledge elements and extend it to capture also the intension of knowledge elements. In honor of Leibniz we call our system ***Leibniz Predicator System*** (LPS) and the numbers for the knowledge units are called ***Leibniz Predicate Numbers*** (LPN) [14]. A similar idea was already formulated by Johnson-Laird [15] but not elaborated. He mentions the set of about 900 words of Ogden´s Basic English [16] *in which it is possible to say just about everything worth expressing, from the works of Shakespeare to modern philosophy*. Also he gives examples of definitions from Longman´s Dictionary of Contemporary English and makes the interesting remark *"the closer the meaning of a word is to some semantically primitive notion, the harder it will be to take its meaning to pieces and to re-express it in terms of other words. It follows conversely that the more complex the meaning of a word, the easier it should be to define"*. The Longmans Dictionary *defines the meanings of 55,000 words using a vocabulary of only 2,000 words.*

The OG in Fig. 5 sketches the main ideas. There are the five basic predicators **>PRE-Thing**, **>PRE-Part, >PRE-not, >PRE-tangible** and **>PRE-living**, which are uniquely identified by the prime numbers **2**, **5, 7, 23** and **29**. The predicators **>PRE-Body** and **>PRE-BodyPart** are derived by the following definitions:

- .Definition (>PRE-Body) = #tangible#Thing#, with
  .LCN (**>PRE-Body**) = .LCN (>PRE-tangible) * .LCN (>PRE-Thing) = 23 * 2 = 46
- .Definition (>PRE-BodyPart) = #Body#Part#, with
  .LCN (>PRE-BodyPart) = .LCN (**>PRE-Body**) * .LCN (>PRE-Part) = 46*5 = 230

The main advantage of using Leibniz characteristic numbers for designating predicators is, that knowledge elements can not only be searched by means of syntactic full text search. Knowledge elements now can also be found by semantic components thereof. In contrast to syntactical search we call this kind of search capability ***Search by Meaning*** (SbM) E.g. the full text search for ***#tangible#* and #thing#** will return **>PRE-Body**. The knowledge element **^BodyPart** can now also be found through its predicator >PRE-BodyPart by searching for the values of its LCNs (.LPN1 = 46 **AND** .LPN2 = 5). Since we also have the .Definition (>PRE-Creature) = #living#Body#, with LCN (**>PRE-Creature**) = .LCN (>PRE-living) * .LCN (>PRE-Body) = 29 * 46 = 1334 = .LPN1 (**>PRE- Creature)** * .LPN2 (**>PRE-Creature**) = 29 * (23 * 2) we can also find *body parts* and *creatures* by their constituting parts. If we want to find concepts which contain the predicator **>PRE-Body**, we can search for predicators which have .LCN = 46 as value for .LPN1 or .LPN2 and we get **>PRE-BodyPart** (.LPN1 = 46) **and >PRE-Creature** (.LPN2 = 46). In other words, if knowledge elements are properly defined in terms of their semantic components, it is possible to retrieve a set of related knowledge elements simply by searching for the LCN. So by the characteristic numbers of Leibniz we get the possibility of ***factorization of knowledge elements*** into more basic knowledge

elements by means of their predicators. Allowing for a semantic decomposition level of 2, a **body part** could also be found by searching for the components **tangible** and **thing**.

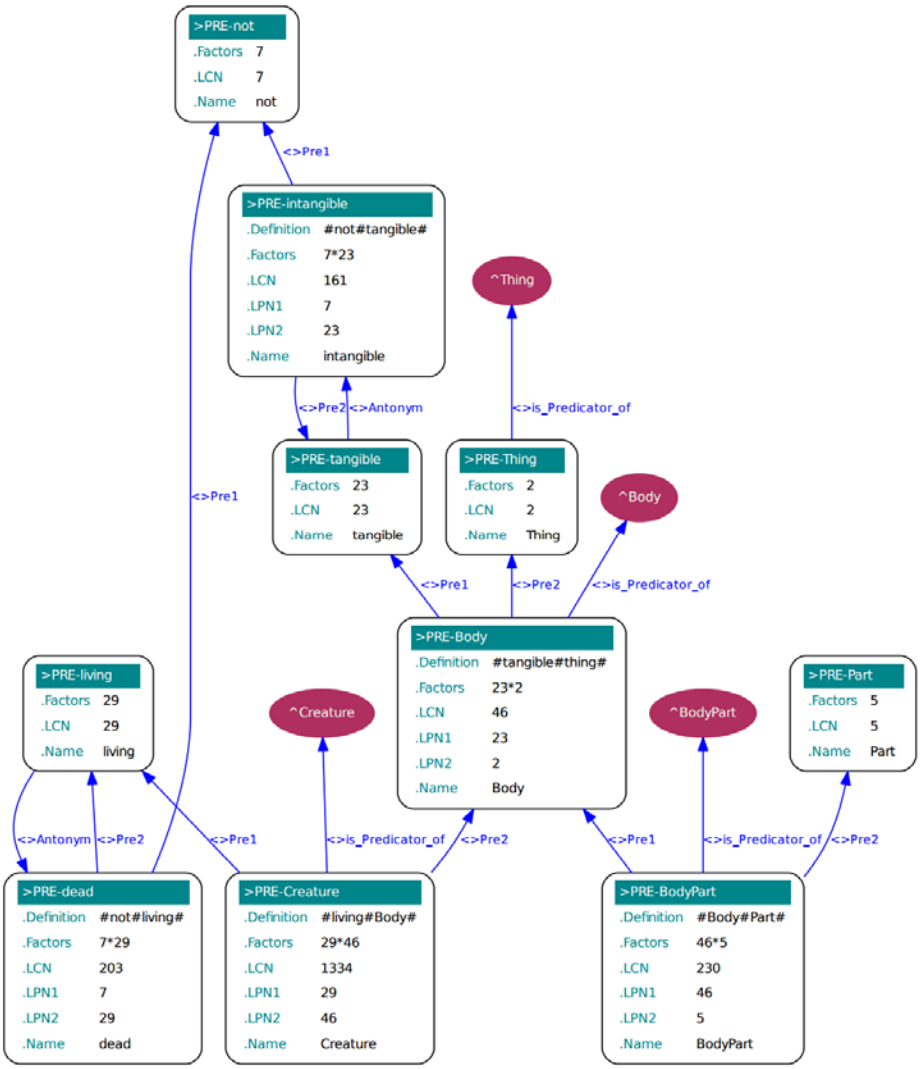



Fig. 5 Definitions of basic knowledge elements in the Leibniz Predication System

## 6. Summary

This paper introduces a proposal for a richer and more precise convention for the naming of knowledge elements and a graphical metaphor for representing the concepts as directed Ontology Graphs (OG). Basic types of knowledge elements are designated by combining special characters with words or proper names like in ^Painting, <>is_Painting_of, ^Painter, ~paint, ^Picasso etc. The choice of special characters for the

naming conventions was mostly guided from practical points of view. So the degrees of freedom where mainly limited by the need of passing predicators as parameters in URLs and by using the predicators as parameters and function names of JavaScript functions. The remaining freedom was used to choose the special characters as close as possible to reflect some kind of mnemotecnic symbolism. For a more precise and better distinction we proposed to include acronyms into the designators like in >**NPS**-Pablo_Picasso (NPS = natural person), >**ATW**_Guernica (ATW = artwork, ^**ANM**_Fish (ANM = animal), ^**ANM** (the class of animals). In the sense of semiotic triangles and for the purpose of disambiguation, the relationships between words and nominators with regard to the designators can be made explicit with relations like (**Picasso**, .is_Lastname_of, >NPS-Pablo_**Picasso**) and (**Picasso**, .is_Name_of, ^**Picasso**). With the LPS (Leibniz Predication System) we presented another method to designate knowledge elements by characteristic numbers and/or their predicators. The class ^Creature is predicated by the relation (>PRE-Creature, <>is_Predicator_of, ^Creature). The predicator >PRE-Creature with .LPN = 1334 = 29 * 46 = 29 * 23 * 2 is composed from the predicators >PRE-living and >PRE-Thing, where >PRE-Thing is a basic predicator, since it is identified by a prime number. By the composition of predicators one can uniquely identify or index knowledge elements, which are semantically constructed from basic predicators. The factorization of predicators allows to retrieve knowledge elements by semantically constituting parts thereof enabling some kind of ***search by meaning***. The relevance of search results now can be additionally controlled by restricting the levels of decomposition of predicators during query execution time.

## 7. References


[1] Knowledge Based Content Management System (KB-CMS) http://www.ontology4.us
[2] Terminology of Knowledge Representations (German) http://schematik.de, last visited: 02/09/2014
[3] Generic Acronyms for Naming of Ontological Concepts http://www.ontology4.us/english/Concepts/Language/Words/Acronyms/, last visited: 02/22/2014
[4] Rudolph Carnap, Meaning and Necessity - A Study in Semantics and Modal Logic, The University of Chicago Press, 1988
[5] Emden R. Gansner,Eleftherios Koutsofios, Stephen North, Drawing graphs with dot, 2009 (Download: http://www.graphviz.org/pdf/dotguide.pdf)
[6] Barry Smith, Against Fantology, 2005 in Johann Christian Marek, Maria Elisabeth Reicher, Experience and Analysis, Wien: öbv&hpt, pp. 153-170
[7] UMO: http://ontology4.us/english/Ontologies/Upper-Ontologies/ UMO%2520Ontology/
[8] Vojtech Svatek, Ondrej Svab-Zamazal, Valentina Presutti, Ontology Naming Pattern Sauce for (Human and Computer) Gourmets, in Eva Blomqvist, Kurt Sandkuhl, Francois Scharffe, Vojtech Svatek, Proceedings of the Workshop on Ontology Pattern, Washington D.C, USA, 2009, pp. 171-178
[9] John F. Sowa, Conceptual Graphs, 2008 in [HaLi2008a] F. van Harmelen, V. Lifschitz, B. Porter, Handbook of Knowledge Representation,Elsevier, pp. 213-237
[10] Frank Manola, Eric Miller, RDF Primer, 2004 http://www.w3.org/TR/rdf-primer/
[11] This Primer is designed to provide the reader with the basic knowledge required to effectively use RDF. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/, last visited: 02/15/2014
[12] RDF is a general-purpose language for representing information in the Web. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/, last visited: 02/15/2014
[13] Wilhelm Kamlah, Paul Lorenzen, Logische Propädeutik - Vorschule des vernünftigen Redens, Verlag J.B. Metzler Stuttgart, Weimar, 1996, ISBN: 978-3-476-01371-2
[14] Designating knowledge elements with Leibniz numbers http://predicator.name, last visited: 03/01/2014
[15] Philip N. Johnson-Laird, Mental Models: Toward a Cognitive Science of Language, Inference and Consciousness, Harvard University Press, 1983, ISBN: 0521273919
[16] OGDEN's BASIC ENGLISH - http://ogden.basic-english.org/words.html, last visited: 03/14/2014
[17] The protégé portal: http://protege.stanford.edu/