„Ethos” 36(2023) nr 3(143) 207-234
DOI 10.12887/36-2023-3-143-12

Jakub GOMUŁKA

ARTIFICIAL INTELLIGENCE APPLIED TO PHILOSOPHY
A Contribution to the Wittgenstein Ontology Project

In 2006, the Wittgenstein Archives at the University of Bergen initiated a project that aimed at encoding Wittgenstein's Nachlass-related information into an explicit symbolic representation conforming Semantic Web technology requirements. However, their ultimate goal is to represent the subject matter of the philosopher's thought. The main purpose of the present paper is to contribute to these efforts.

For a few centuries, the word "ontology" has been used in multiple ways: a branch of philosophy that deals with what exists; a fundamental categorisation of reality; an attempt to understand being itself; or a structure of the most general possibilities. However, in recent decades, it has gained a new meaning in the context of computer science and, more specifically, in what is called Knowledge Representation and Reasoning (KRR or KR^2). KR^2 is an offshoot of the symbolic AI research programme (also dubbed GOFAI—"good old-fashioned AI") that was initiated in the 1950s by a group of scholars that included Allen Newell, Herbert A. Simon, and Marvin Minsky. This current dominated the AI field until around the middle of the 1990s, but it has since then been gradually ousted by the artificial neural networks paradigm that is nowadays almost synonymous with AI research as such. The proponents of symbolic AI believed that the ultimate goal of their discipline—creating a machine that would possess the ability to think like a human being (in other words, artificial general intelligence)—can be achieved by developing declarative high-level symbolic representations of problem domains as well as formal rules of reasoning on the basis of such representations. As we know today, the initial high expectations were not met due to the problem of scaling algorithmic solutions from simple toy problems to their real-life applications. Nevertheless, the knowledge representation approach was applied in useful ways in the 1970s and 1980s in the form of expert systems and still serves many purposes in both commercial and non-commercial web services and in ongoing AI research programmes such as intelligent agent approach. One offspring of the expert system paradigm is Semantic Web technology: a set of standards for data encoding, machine reasoning, and the representation of knowledge in the Internet.

To build a useful declarative representation of a piece of knowledge, one needs a shared conceptual model of a domain they want to represent; in other words, a formal explicit specification of a conceptual structure for that domain that can be known to potential users of that representation. Such a specification is called "ontology"[1].

[1] See Thomas R. G r u b e r, "Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition* 5, no. 2 (1993): 199.

In a broader sense, the term can be attributed to a whole knowledge base structured by a given conceptual model. In order to separate this meaning of the word from its philosophical use, in the present paper, we shall instead use the composite term "computational ontology." Several different types of computational ontologies can be singled out: there can be general purpose conceptual models that are relatively domain-independent and include such universal categories as space or time; structures of categories that are designed to represent a specified domain or task; common sense models that serve to capture general knowledge; and also metadata ontologies that specify the vocabulary used to describe the structure of data available online.[2]

In 2006, the Wittgenstein Archives at the University of Bergen (WAB), an institution whose primary task is to make the Wittgenstein *Nachlass* available for researchers all around the world, initiated a project that aimed at encoding the *Nachlass*-related information into an explicit symbolic representation conforming Semantic Web technology requirements.[3] Consequently, the WAB started developing a computational ontology that currently organises information about over fifty thousand *Nachlass* remarks and fragments of Wittgenstein's published books; provides them with a link to a specific page of Wittgenstein's manuscript, typescript, or publication; relates them to the persons they mention; and annotates them with the dates when they were written. The knowledge base thus created is available and easily searchable at the WAB website.[4] It can be said that the project, called "Wittgenstein Ontology," already provides metadata that can be very useful for scholars interested in the development of Wittgenstein's philosophy. However, the aims of the WAB team[5] are much more ambitious: their ultimate goal is to represent the subject matter of Wittgenstein's thought.[6] There are several reasons why such a task is difficult to accomplish: the current

[2] See Diana Marcela S á n c h e z, José María C a v e r o, and Esperanza M a r c o s M a r t í n e z. "The Road Toward Ontologies," in *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems*, ed. Raj Sharman, Rajiv Kishore, and Ram Ramesh (New York: Springer, 2007), 9n.

[3] See Alois P i c h l e r, *Wittgenstein Ontology* (Bergen: University of Bergen, 2013), http://wab.uib.no/wab_philospace.page.

[4] Wittgenstein Ontology Explorer, http://wab.uib.no/sfb/.

[5] In the present paper, the term "WAB team" is used in a loose sense: in fact, it means a group of people involved in Wittgenstein ontology research who co-authored papers devoted to the project. In other words, it is comprised of Alois Pichler (the longtime director of the WAB), Amélie Zöllner-Weber, Jakub Mácha, Rune Falch, Matthew Fielding, Nivedita Gangopadhyy, Andreas Opdahl, Øyvind Liland Gjesdal, and a few others. Note that this list does not include all the researchers affiliated with the WAB and includes individuals who are not and never were formally affiliated with the institution.

[6] See Alois P i c h l e r and Amélie Z ö l l n e r-W e b e r, "Sharing and Debating Wittgenstein by Using an Ontology," *Literary and Linguistic Computing* 28, no. 4 (2013): 700–707; Jakub M á c h a, Rune F a l c h, and Alois P i c h l e r, "Overlapping and Competing Ontologies in Digital Humanities," in *DH-CASE '13: Proceedings of the 1st International Workshop on Collaborative Annotations in Shared Environment; Metadata, Vocabularies and Techniques in the Digital Humanities (10 September*

technology and tools that comprise Semantic Web paradigm seem to be inadequate for the representation of knowledge in the humanities. The main purpose of the present paper is to contribute to the WAB's efforts to build a comprehensive representation of Wittgenstein's philosophical legacy. All the examples discussed in the paper come from the *Tractatus Logico-Philosophicus* translated by Pears and McGuinness.[7]

We will begin with a brief outline of the basic components of the existing technology: RDF, OWL2, and SPARQL. Particularly, we will focus on the RDF reification mechanism that can prove useful in dealing with the problem of inconsistencies and different interpretations of knowledge in the humanities. Next, we will discuss the main difficulties that researchers encounter as they attempt to represent philosophical conceptions within the current paradigm. Subsequently, we will turn to the conceptual structure proposed by the WAB team, discuss the possibility of extending this structure, and try to apply WAB ideas regarding perspectives, claims, and concepts. In the course of the latter, the representation of some actual Tractarian sentences will be discussed. Particular attention will be paid to symbolic formulas. The paper concludes with an outline of possible developments of the KR^2 technology and their importance for the AI programme.

AN OVERVIEW OF THE ELEMENTS OF EXISTING SEMANTIC WEB TECHNOLOGY

Semantic Web (SW) aims to represent information that has a well-defined structure and can be easily searchable for classic algorithmic systems. The information stored according to the SW paradigm allows for complicated and precise query, automated processing, and generating additional information that has not been explicitly provided. The paradigm implements the following general principles:

(1) sharing structured data in the form of documents written in a commonly accepted notation;

(2) representing information articulated in entities with specified properties;

(3) sharing machine-readable formal descriptions of data semantics.

The three principles are implemented, firstly, by using IRIs—Internationalised Resource Identifiers—that are usually URLs (uniform resource web addresses), to identify basic objects called resources that constitute represented knowledge. Secondly, it does so by formatting information according to the Resource Descrip-

2013, Florence, Italy), ed. Francesca Tomasi and Fabio Vitali (New York: Association for Computing Machinery, 2013); Alois P i c h l e r et al., "Crisscross Ontology: Mapping Concept Dynamics, Competing Argument and Multiperspectival Knowledge in Philosophy," *Quaderni di "Filosofia,"* no. 2 (2021): 59–73.

[7] See Ludwig W i t t g e n s t e i n, *Tractatus Logico-Philosophicus*, trans. David F. Pears and Brian F. McGuinness (London and New York: Routledge and Kegan Paul, 1965).

tion Framework (RDF) standard in named graphs that describe relations between entities[8]. The third way the principles are implemented is serialising named graphs in one of several and mutually exchangeable formats (Turtle, N-Triples, N-Quads, JSON-LD, RDF-XML, or RDF-JSON). Lastly, this is accomplished by using formal languages to define vocabularies used in RDF descriptions; in other words, computational ontologies that consist of hierarchies of classes and properties, restrictions, and rules of generating new information from that which is given. Currently, the most popular standard for defining ontologies is OWL2[9] combined with RDF Schema 1.1, which is an extension of RDF that was introduced in 2014.[10] The most common software tool for managing information stored in SW knowledge bases is SPARQL.[11]

The simplest piece of information represented in an RDF graph is an RDF triple that consists of three members: a subject, a property, and an object. The first of these is a resource to which the property is attributed; the last, meanwhile, can be understood as the property's value. The role of the RDF subject can be taken by an entity: a type of a resource (with a unique IRI identifier) that can stand in numerous different triples both on the position of subject and object. The role of the RDF property must be taken by a property: another type of a resource. Entities can be subsumed to types or classes by a predefined property "type," and properties can be ordered in a hierarchy. The basic structure of the triple is presented in Diagram 1.

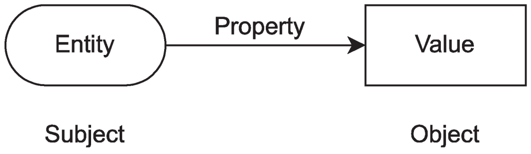

Diagram 1. The structure of the RDF triple.

There are two general types of RDF properties that differ in the kinds of values they can take. Data properties take as their values strings of Unicode characters that can have one of many predefined formats, including: date and time stamps, floating-point numeric values, or ISO language codes. Strings are not resources, and they are

[8] See RDF Working Group. *Resource Description Framework (RDF)*. W3C. February 25, 2014. https://www.w3.org/RDF/.

[9] See W3C OWL Working Group. *OWL 2 Web Ontology Language*. W3C. December 11, 2012. https://www.w3.org/TR/2012/REC-owl2-overview-20121211/.

[10] See Dan B r i c k l e y and R.V. G u h a, *RDF Schema 1.1. W3C Recommendation*, February 25, 2014, https://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[11] See Eric P r u d' h o m m e a u x and Andy S e a b o r n e, *SPARQL Query Language for RDF, W3C Recommendation*, January 15, 2008, https://www.w3.org/TR/rdf-sparql-query/.

not identified by IRIs, so even if we put the same string as a value in several triples, each instance of the string will be treated as something separate.

Object properties take RDF entities as their values; in other words, object properties can be seen as binary relations between two entities. As we know, binary relations can have such features as symmetricity, reflexivity, or transitivity. All such features can be attributed to properties because the latter can take the role of the RDF subject as well. It should be noted that both object and data properties can be attributed to one entity with different values several times: an entity "Alice" can have several values of an object property "hasChild." However, it is possible to limit a property to just one value by making it functional.

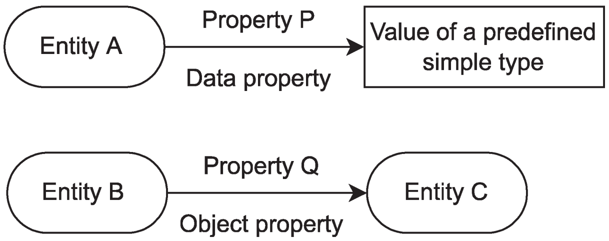

Diagram 2. Data and object properties.

We can illustrate the use of both data and object properties with examples related to Wittgenstein's *Tractatus*. "*Tractatus Logico-Philosophicus* has 526 theses" can be represented by the entity "Tractatus Logico-Philosophicus," the data property "number of theses," and the simple integer value "526" (see Diagram 3).

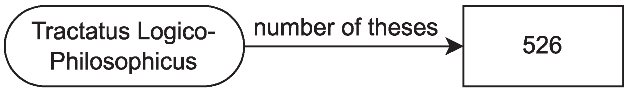

Diagram 3. An example of a data property.

In turn, the first sentence of the *Tractatus* 6.031: "The theory of classes is completely superfluous in mathematics" can be represented by the object property "redundant in" attributed to the entity "Theory of classes" with the entity "Mathematics" as its value (see Diagram 4).

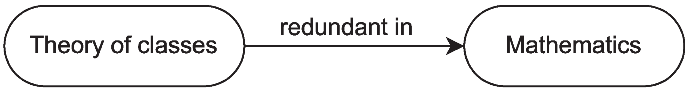

Diagram 4. An example of an object property.

The two graphs presented above can be serialised in Turtle notation as follows:

```
@prefix ex: <http://example.org/test#> .

ex:Tractatus_Logico-Philosophicus ex:number_of_theses “526” .
ex:Theory_of_classes ex:redundant_in ex:Mathematics .
```

As we can see, RDF triples usually end with a dot. The initial triple does not encode any graph; it sets a prefix to the names of all our resources used later in the code. The prefix stores a base URL address that supplements a particular resource name. In the above example, the prefix is called ‘ex’ (this name has been arbitrarily chosen) and stands for the fictitious URL address “http://example.org/test#” (it would be better if real SW knowledge bases used real URL addresses). Therefore, the actual IRI for the entity “Tractatus Logico-Philosophicus” that occurs in the second triple is “http://example.org/test#Tractatus_Logico-Philosophicus.”

Certainly, there are plenty of sentences whose structure cannot be projected into simple RDF triples without a significant loss of meaning. Fortunately, RDF specification envisages a feature that makes creating much more syntactically complex representations possible: a blank node which is not a resource and does not have an IRI. Blank nodes can stand in place of both triple’s subject and object. Therefore, they can be used to create tree-like structures that represent a more complicated syntax of a given sentence (see Diagram 5).

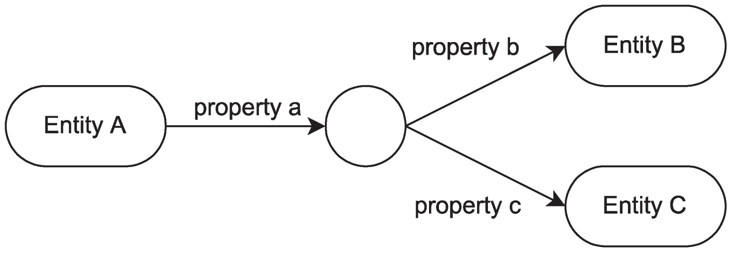

Diagram 5. A blank node structure.

For instance, if we want to represent the sentence “John travels to Vienna by train,” we can use one blank node that can represent travel (although it is possible, we do not need to name a blank node). Thus, the first RDF triple would represent the part of the information carried by the sentence; namely, that John travels. Subsequently, the blank node can have properties that specify the direction of the travel and means of transportation; therefore, the second and third RDF triples would indicate that the travel is to Vienna and that it is done by train (see Diagram 6).

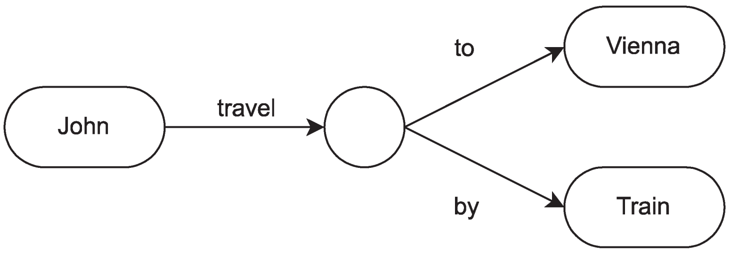

Diagram 6: A blank node example.

The Turtle symbol for a blank node is square brackets. In our example, the square brackets occur as an object in the first triple. Then, the second and third triples, whose subject is the blank node itself, appear separated by a semicolon within the brackets without their first member: by default, their subject is the blank node.

```
ex:John ex:travel [ ex:to ex:Vienna; ex:by ex:Train ] .
```

Equipped with this feature, we can deal with such complicated sentences as the first sentence of the *Tractatus* 6.22: "The logic of the world, which is shown in tautologies by the propositions of logic, is shown in equations by mathematics." We can attribute the object property "shown" to the entity "Logic of the world" twice: both times the objects would be blank nodes that, in turn, have their own pairs of object properties: "by" and "in." The two blank nodes represent the two aspects of showing; the first related to tautologies and propositional logic, the second to equations and mathematics (see Diagram 7).

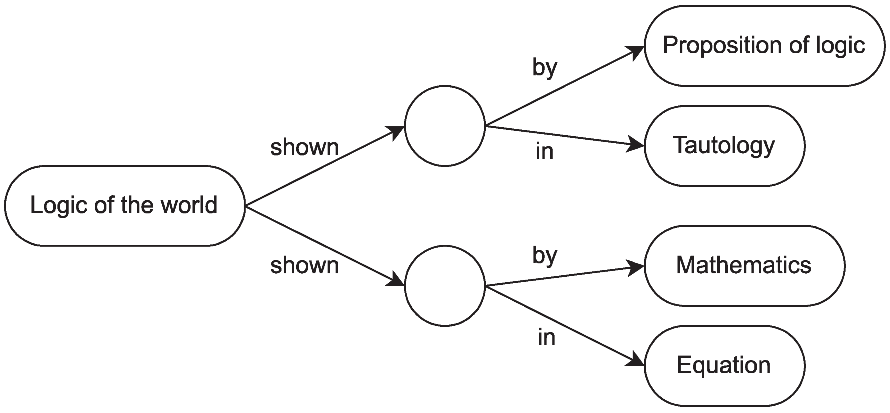

Diagram 7: A named graph for the first sentence of the *Tractatus* 6.22.

The Turtle notation for the above graph is as follows (indents and new lines are only introduced to make the code clearer; they are optional elements of syntax):

```
ex:Logic_of_the_world ex:shown
[ ex:by ex:Proposition_of_logic; ex:in ex:Tautology ],
[ ex:by ex:Mathematics; ex:in ex:Equation ] .
```

As the code makes explicit, if we have two triples that share the same RDF subject and RDF property, and differ only at the RDF object, we can serialise them as one triple with both values separated with a coma.

As we will see later in this paper, from the perspective adopted in the humanities, one of the most interesting features of RDF specification is so-called reification: some RDF entities can represent RDF triples themselves. They possess special predefined RDF properties: “subject,” “predicate,” and “object.” The values of these properties can be corresponding members of a triple.[12]

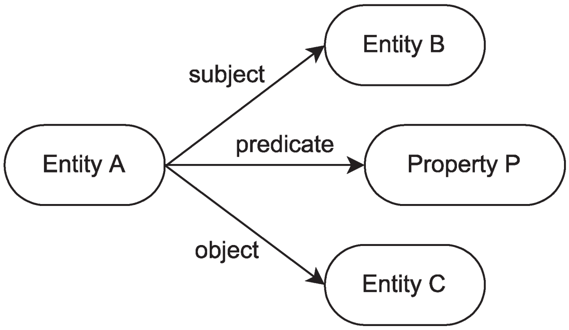

Diagram 8. The idea of RDF reification.

We can illustrate the RDF reification mechanism with another example taken from *Tractatus Logico-Philosophicus*; that is, the first sentence of thesis 6.2341: “It is the essential characteristics of mathematical method that it employs equations.” The triple that would be a subject to reification is: “mathematical method employs equations.” This fact is, as the original sentence states, the essential feature of mathematical method; consequently, the non-reified triple would consist of mathematical method as a subject, essential feature as a property, and the reified statement as an object (see Diagram 9).

[12] See Dean A l l e m a n g, Jim H e n d l e r, and Fabien G a n d o n, *Semantic Web for the Working Ontologist. Effective Modeling for Linked Data, RDFS, and OWL* (New York: Association for Computing Machinery, 2020), 53–56.

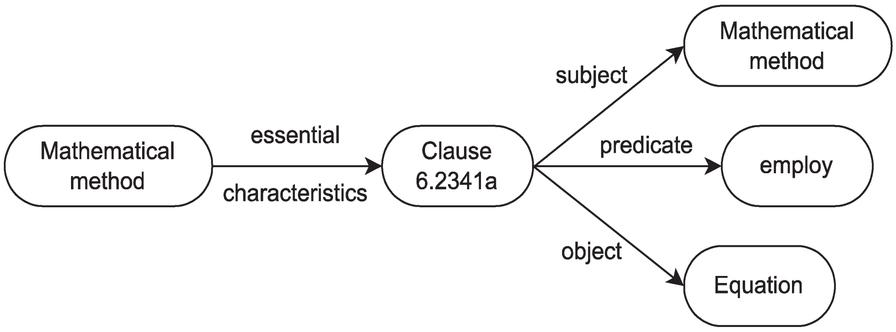

Diagram 9. An application of RDF reification

The Turtle serialisation of the above graph requires the introduction of yet another prefix because we make use of RDF predefined resources:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

ex:Mathematical_method ex:essential_characteristics ex:Clause_6.2341a .
ex:Clause_6.2341a rdf:type rdf:Statement ;
            rdf:subject ex:Mathematical_method ;
            rdf:predicate ex:employ ;
            rdf:object ex:Equation .
```

Each entity we introduce to our knowledge base can be of a certain type. In the above example, the entity “Clause_6.2341a” is a type of “Statement.” The latter is an RDF predefined class that is used for reification. However, we can easily create our own classes by declaring entities of the type “Class,” which is a predefined element of the OWL syntax. The fact that classes are also entities, and therefore resources, as well as the fact that the predefined element “Class” is itself both a resource and a class, may appear a bit confusing, but this does not lead to any real ambiguities. “Class,” together with the RDF Schema predefined property “subClassOf,” allow us to create a hierarchical categorisation; that is, a conceptual ontology, as in the following example:

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix wab: <http://purl.org/wittgensteinsource/ont/scholar/0.1/> .

wab:Subject rdf:type owl:Class .
wab:Source rdf:type owl:Class .
wab:Perspective rdf:type owl:Class ;
          rdfs:subClassOf wab:Subject .
wab:Primary_Source rdf:type owl:Class ;
```

```
            rdfs:subClassOf wab:Source .
wab:WittgensteinSource rdf:type owl:Class ;
              rdfs:subClassOf wab:Primary_Source .
```

As can be guessed from the prefixes, this is a part of a real project; that is, the Wittgenstein ontology that has been developed by the WAB.

There are numerous ways to define classes and properties in OWL, but we will not discuss this topic here. However, we should point to certain mechanisms that play their roles in the automatic production of new knowledge. Firstly, for a given property it is possible to determine the type of entities that are subjects of that property as well as the types of its values. For instance, if we define the property "is sentence of" as follows:

```
ex:isSentenceOf rdf:type owl:ObjectProperty ;
          rdfs:domain wab:Sentence ;
          rdfs:range wab:NachlassBemerkung .
```

then we will know that each entity to which that property is attributed is of the type "Sentence," and each entity that is a value of that property is of the type "Nachlass Bemerkung." Secondly, object properties can have special meta-properties that determine whether they are transitive, reflexive, irreflexive, symmetric, asymmetric, and functional and whether they are inverses of some other object properties. For instance, if we attribute the OWL predefined meta-property "TransitiveProperty" to some object property that links an entity A with an entity B and an entity B with an entity C, then we will know that it also links an entity A with an entity C. Thirdly, we can determine whether two or more distinct classes are mutually disjointed; namely, whether there is no single entity that is a member of more than one of them. So, if we ascribe a given entity to one of the group of mutually disjoint classes, we will know that the entity does not belong to any of the others.

The additional knowledge mentioned in the paragraph above can be obtained automatically as we apply a software device called a reasoner. A working reasoner extends our knowledge base with additional automatically computed RDF triples. It is worth noting that our ontologies can prove incoherent: one or more additional triples can turn out inconsistent with the existing ones. The expressivity of OWL2—much richer than that presented above—is somewhat lesser than the expressivity of the first order calculus in order to keep the reasoning process within the SW knowledge base decidable. We want to know for certain if our ontology is coherent and which additional RDF triples are computed.

SW knowledge bases would be useless if there was no tool to retrieve and manipulate the information stored in them. Just as the purpose of SQL (Structured Query Language) is the management of data held in relational databases, likewise SPARQL

(SPARQL Protocol and RDF Query Language) is a tool to manage data stored in RDF triples and structured by OWL ontologies. The most typical task for SPARQL queries is to retrieve information from a knowledge base. It can be done, just as in SQL, with the "SELECT" command followed by an indication of what should be retrieved and the 'WHERE' instruction followed by a clause that narrows the search placed in curly brackets. A WHERE-clause may contain an RDF graph pattern to which all results must conform. Turtle notation can be used to describe graphs, so the query can be formulated as follows:

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

SELECT ?p WHERE { ?p rdf:type owl:Class . }
```

A graph pattern is created when we replace at least one of the triple's members with a variable whose name begins with the "?" sign. The query presented above will produce a list of all knowledge base entities that are classes.

We can also compose much more complicated queries by combining two or more graph patterns together and linking them with variables that do not need to occur in a SELECT-clause. Therefore, queries can be as precise as needed, and their results can be adequate. This possibility was one of the original justifications for the development of the SW technology: it seemed that the technology could provide a means to navigate effectively over the vast and chaotic collection of information that makes up the Internet. The pertinence of this line of justification faded with the rise of web search engines like Google,[13] but it should be noted that at time of writing Google algorithms still fail to deliver adequate results in case of more complicated queries.

This brief presentation of the elements of the SW technology is far from exhaustive. Many of its elements have not been mentioned at all, and those mentioned have been discussed very concisely. Therefore, one should not treat this chapter as a comprehensive introduction to the topic but rather as a necessary context for proposals that will be discussed later.

PHILOSOPHICAL KNOWLEDGE
THE HARD PROBLEM OF KNOWLEDGE REPRESENTATION?

The SW technology is best suited for representing definite knowledge. Imagine having a collection of some objects that have easily discernable features and can be

[13] See Catherine C. M a r s c h a l l, and Fran M. S h i p m a n, "Which Semantic Web?", in *HYPERTEXT'03: Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia 2003*: 58f., https://dl.acm.org/doi/proceedings/10.1145/900051.

grouped in a clear hierarchy of classes. The knowledge about that collection can be represented without any problem; it would be a kind of catalogue. A good example of such a catalogue is the current state of the Wittgenstein ontology project: so far, the WAB team has produced over five hundred thousand RDF triples that involve over sixty-five thousand individual entities, including over fifty thousand entities for each separate remark by Wittgenstein. The remaining several thousand entities stand for persons, dates and periods of time, volumes or books, and others. They are linked together by twenty-six object properties, including "refersTo," "hasDate," or "hasPart," as well as several data properties.[14]

Unfortunately, representing a theory, in particular, a philosophical conception is a different story. There are numerous reasons why such a task is highly problematic. The first and most important reason is the inherent multi-perspective nature of the humanities.[15] The same phenomenon prompts different thinkers to present their own accounts; moreover, those accounts, in turn, prompt other scholars to present their own readings of these different accounts, bringing about new layers of divergence. Although debates are generally possible, they usually result in the formulation of yet more theories and interpretations. There is nothing wrong in this process; this is how knowledge in the humanities progresses. However, there is no clear solution to the problem of how to represent such knowledge. What are to be the entities and properties in a situation when various perspectives can offer completely different categorisations of a given phenomenon?

The second problem is the contextuality and indefiniteness of conceptions in the humanities. There is nothing strange or unnatural in the existence of their various interpretations: they are structurally open to complementing with new content and for employing in various situations that change their meanings. Therefore, knowledge in the humanities (philosophy) can never reach its definite shape and ultimate interpretation.

There is also a problem of inconsistency: some standpoints in the humanities are plainly inconsistent; that is, they contain two or more inconsistent claims. Meanwhile, others are inconsistent because it is possible to reach inconsistency through inference. Moreover, they can be inconsistent in various ways. For example, the notorious *Tractatus* 6.54 says that to understand its author is to recognise the Tractarian theses as nonsensical, as something one should throw away like a ladder. How to represent this? Is there a possibility to create a coherent computational ontology for Wittgenstein's *Tractatus* or Hegel's *Phenomenology of Spirit*?

Finally, the humanities are often meta-theoretical: various stances include specific categorisations of the domains about which they theorise. It is tempting to try to represent such categorisations as computational ontologies of OWL classes, but such attempts are doomed to fail because philosophical conceptual structures do not meet the formal

[14] See Alois P i c h l e r and Øyvind Liland G j e s d a l, W*ittgenstein Ontology* (Bergen: University of Bergen, 2007), http://ubbdev.gitlab.io/wab-ontology/index-en.html.

[15] It should be noted that both this and other features of knowledge in the humanities (philosophy) mentioned in the main text can also be attributed, perhaps to a lesser extent, to scientific theories.

strictness requirements of first order calculus. Moreover, categorisations in the humanities are partly grounded in implicit relations between basic concepts that are informal, semantic in nature. Thus, if primitive concepts of ontologies in the humanities cannot be mapped directly onto OWL classes, what type of entities should they be? In other words, how can one build a computational ontology for a philosophical ontology?

The WAB team has long been aware of those difficulties, and they have made attempts to address them both in their theoretical papers and in the actual shape of the Wittgenstein ontology. They tested and subsequently rejected the idea of mapping philosophical categorisation onto the hierarchy of OWL classes.[16] They also admitted both the possibility that various philosophical claims of the same philosopher may contradict each other, as well as the possibility that the same philosophical content may be interpreted as various conceptual structures.[17] Furthermore, they have described philosophical content as dynamic, open-ended, vague, and context-dependent.[18]

The WAB answer to the problem of representing philosophical ontologies is the flat and limited hierarchy of classes that are directly responsible for grouping content-related entities. In turn, their solution to the other three problems is to introduce a class "Perspective" and make representations of the philosophical content dependent on it. Diagram 10 illustrates the current state of the Wittgenstein project class hierarchy. It consists of three top-level classes: "Person," "Source," and "Subject." The first of them groups entities that represent persons pertinent to the content of the Wittgenstein *Nachlass* and has no child-classes. The "Source" class is the most ramified: its direct child-classes are "Primary Source" and "Secondary Source." The former has four child-classes, including the "WittgensteinSource" class that, in turn, has nine child-classes. Those bottom-level classes are used to group such Nachlass-related entities as separate remarks ("Nachlass Bemerkung", "Part"), manuscripts ("MS"), and typescripts ("TS").

The third top-level class, "Subject," has eight child-classes that constitute the bottom level of that branch.[19] Four of them, "Language," "Date," "Place," and "TextSub-

[16] Amélie Z ö l l n e r-W e b e r and Alois P i c h l e r, "Utilizing OWL for Wittgenstein's Tractatus," in *Papers of the 30th International Ludwig Wittgenstein Symposium (5–11 August 2007, Kirchberg am Wechsel),* ed. Herbert Hrachovec, Alois Pichler and John Wang (Kirchberg am Wechsel: ALWS, 2007).

[17] See Jakub M á c h a, Rube F a l c h, and Alois P i c h l e r , "Overlapping and Competing Ontologies in Digital Humanities," in *DH-CASE '13: Proceedings of the 1st International Workshop on Collaborative Annotations in Shared Environment: metadata, vocabularies and techniques in the Digital Humanities (10 September 2013, Florence, Italy),* ed. Francesca Tomasi and Fabio Vitali (New York: Association for Computing Machinery, 2013).

[18] See P i c h l e r, F i e l d i n g, G a n g o p a d h y a y, and O p d a h l, "Crisscross Ontology: Mapping concept dynamics, competing argument and multiperspectival knowledge in philosophy," 63.

[19] We have set aside the two classes the WAB team proposed in their most recent paper (see ibidem, 70) that are not yet present in the most recent version of the knowledge base (namely, "Debate" and "Argument"), although we are far from rejecting their usefulness. This only means that they are not pertinent to the proposals discussed in the present paper and that our primary reference is the knowledge base itself.

Genre," are not pertinent to the philosophical content but rather to some kinds of facts regarding the documents described in the "Source" branch. The fifth, "Field," refers to the philosophical subject matter in a very general way: it is designed to group entities that represent specific philosophical subdisciplines (currently, it includes two such entities: "Ethics" and "Philosophy of Language"). The remaining three classes that are crucial for representing philosophical knowledge are: the aforementioned "Perspective," "Point" (or "Claim"), and "Issue" (or "Concept"). The alternative names for the last two classes indicate that the project is undergoing a transitive period regarding class naming: the most recent version of the Wittgenstein ontology OWL file,[20] available on December 14, 2022, still uses older versions (namely, "Point" and "Issue"), while the most recent publications use both old and new names[21]. Nowadays, meanwhile, WAB scholars are more inclined to use "Claim" and "Concept" when discussing the project.

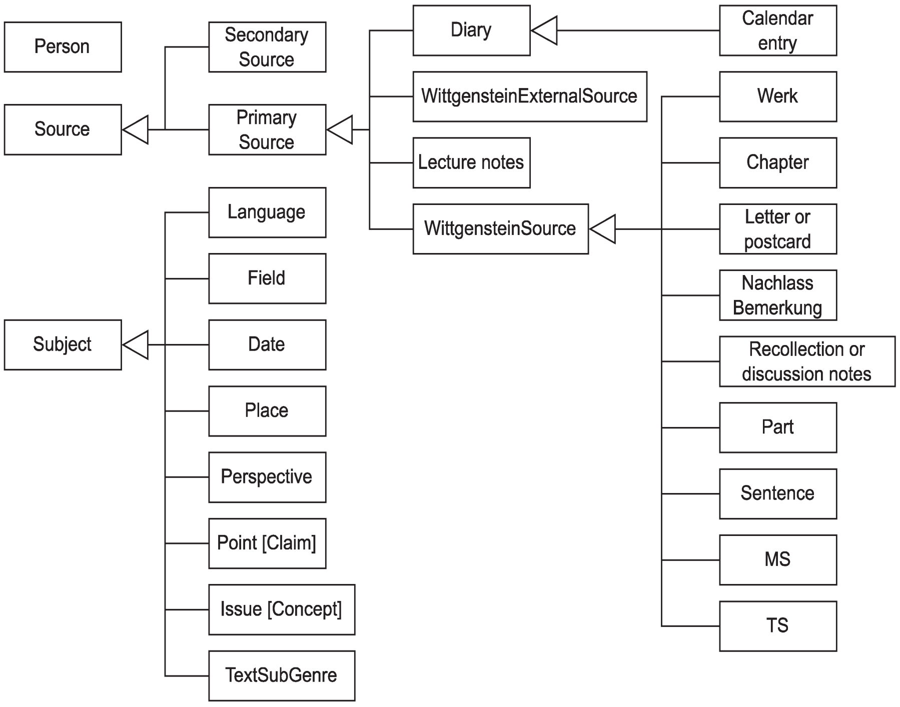

Diagram 10. The Wittgenstein ontology class hierarchy (the arrows point at the parent-classes of particular child-classes).

[20] The file is available for download from the web address: Wittgenstein Ontology Explorer, http://wab.uib.no/cost-a32_philospace/wittgenstein.owl (due to the large size of the file, downloading it may require a significant amount of time).

[21] See P i c h l e r, F i e l d i n g, G a n g o p a d h y a y, and O p d a h l, "Crisscross Ontology: Mapping Concept Dynamics, Competing Argument and Multiperspectival Knowledge in Philosophy," 59–73; P i c h l e r and G j e s d al, *Wittgenstein Ontology*.

However, it is not quite clear how exactly the Wittgenstein ontology developers imagine relations between the exponents of the three classes; particularly, how claims and concepts will relate to perspectives. The 2021 paper suggests that node-structures for “Point” (“Claim”) and ‘Concept’ members should be developed separately: it focuses more on relations within both classes; that is, relations between different concepts and relations between different claims rather than those that link claims with concepts. It seems that the latter are simply instances of a single type of relation of occurrence of a concept in a claim.[22] Diagram 11 illustrates what points (claims) and concepts are meant to represent.

It is particularly unclear what the authors mean when they write that “concepts and points shall both be allowed to develop their own sub-ontologies within the overarching general domain ontology”[23]. The context suggests that this has something to do with the entities grouped in the “Perspective” class. The paper’s authors continue to write that “one will be able to merge graphs for different points which use the same concepts and are non-contradictory into larger graphs which, consequently, can represent an actual viewpoint within the specific domain,” and such larger graphs are to be perspectives: groups of non-contradictory statements by a given thinker or scholar.[24] However, the authors do not specify how they wish to associate an entity with a complex graph. Moreover, they make no specific remarks regarding the relation of concepts (and their interrelations) to perspectives: it seems as if a given domain of concepts is meant to be linked with a certain perspective only via a certain domain of claims.

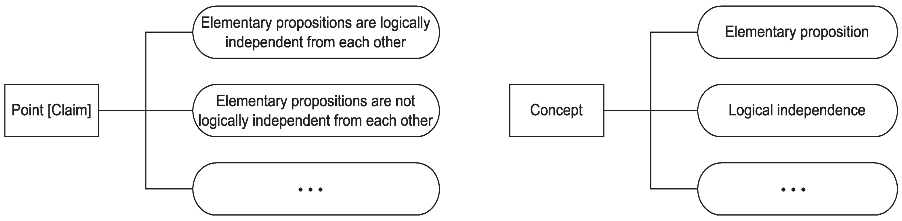
Point [Claim]
Elementary propositions are logically independent from each other
Elementary propositions are not logically independent from each other
…
Concept
Elementary proposition
Logical independence
…

Diagram 11. The “Point” (“Claim”) and “Concept” classes (based on: P i c h l e r, F i e l d i n g, G a n g o p a d h y a y, and O p d a h l, “Crisscross Ontology: Mapping Concept Dynamics, Competing Argument and Multiperspectival Knowledge in Philosophy,” 62).

[22] P i c h l e r, F i e l d i n g, G a n g o p a d h y a y, and O p d a h l, “Crisscross Ontology: Mapping Concept Dynamics, Competing Argument and Multiperspectival Knowledge in Philosophy,” 66–69.

[23] Ibidem, 69.

[24] Ibidem.

The 2021 WAB team paper puts forward an outline of their general task: to develop a SW knowledge base that would meet the needs of the humanities; that is, “to design novel approaches to ontology design, so that these can fully integrate humanities and philosophy contents while at the same time still retain the traditional strengths and assets of ontology work such as formal precision, cognitive economy, maximum interoperability and explanatory power, as well as permitting standard querying and inference tasks”.[25]

In the concluding remarks, the authors admit that they have not yet figured out how exactly such a task can be accomplished.[26] In the next section of this paper, we will attempt to provide some detailed partial solutions. The solutions may miss the aim to retain all the traditional assets of computational ontology, but this should not be considered their flaw but rather their advantage.

PERSPECTIVES, CLAIMS, CONCEPTS, RULESETS, AND SYMBOLS

Let us commence with a couple of remarks regarding the class hierarchy proposed by the WAB. It is generally worth preserving if only for the reason that much very useful work has already been done, and the computational ontology in question is inextricably linked to the vast amount of data of which the Wittgenstein ontology project is comprised. Moreover, the approach that leaves the “Subject” branch as flat as possible seems to be right.

One of the mechanisms that will be discussed in this chapter requires a new “WittgensteinSource” child class that would group entities representing symbolic for mulas used by Wittgenstein in some of his sentences. The most natural name for such a class is “Symbol,” however, the name is a bit misleading in the context of the *Tractatus*.[27] By “symbolic formulas,” we are referring to all strings of characters that are meant to figure in formal transformations according to a certain calculus. We can find a significant number of such strings in the *Tractatus* alone: the “$[\bar{p}, \bar{\xi}, N(\bar{\xi})]$” expression from thesis 6, “$(\exists x): aRx.xRb$” from thesis 4.1252, and also “0 + 1 = 1 Def.” from thesis 6.02. The exponent of the “Symbol” class is just a node in the knowledge base; its name can be arbitrary, although it would be practical if it followed the naming convention applied in the “Source” branch and marked the volume and page of its occurrence. It should have a data property that would be similar to the proposed “shape” data property for the “Sentence” class. In fact, it can be the very same property defined for both classes. Regarding its value, it is much more convenient to use a richer notation than just a Unicode plain text, such as a very popular TEX syntax. Accordingly, the values of the “shape” property of “Symbol” entities for the formulas presented above would be:

[25] Ibidem, 65.

[26] Ibidem, 71.

[27] The early Wittgenstein uses two terms: “sign” and “symbol.” A sign is a set of marks of a given shape that have a syntax and a symbol is a sign with meaning. Thus, what we understand by the term “symbol” is actually the Tractarian “sign.”

“[\overline{p}, \overline{\xi}, N(\overline{\xi})],’ ‘(\exists x):aRx.xRb,’ ‘1 \coloneqq 0 + 1.”

The decision to place the “Symbol” class within the “Source” branch is only provisional because it is not quite obvious that this should be its final place. On the one hand, “Symbol” members are to represent real pieces of source textual material; moreover, they should be linked to “Sentence” exponents with the aforementioned property “is part of” that, in turn, should operate within the “Source” branch only. On the other hand, as we will see, “Symbol” entities are meant to figure in subject-matter conceptual structures: symbols belong to the abstract content rather than to the textual, material form. Moreover, the “Symbol” members would play a crucial role in the aforementioned new mechanism that plainly belongs to the “Subject” domain: a mechanism to engage parts of the represented philosophical content as elements of new inference rules used by a system to generate additional information. We will discuss the details of the mechanism near the end of this section of the paper. For the moment, it should only be mentioned that it would require yet another extension of the class hierarchy: a new “Subject” child class called “Ruleset.”

To summarise the remarks on the current Wittgenstein ontology class structure, we can say that it should generally remain in the present form, and its future extensions should be rather limited. The two additional classes proposed above—“Symbol” and “Ruleset”—are strictly related to the extension of the existing SW technology that is required by the new mechanism we want to introduce.

Before we delve into more detailed presentations of the possibility of binding perspectives, claims, and concepts together, let us point to two general facts. Firstly, we should differentiate two kinds of entities of which our knowledge base would consist: one would be the resources that are employed to directly represent the elements of the knowledge domain, while the other would be those that categorise, organise, and link the former. In other words, we have the subject-matter level resources and the meta-level resources. All classes in our knowledge base would work as meta-level elements, while the vast majority of entities that are not classes would be subject-matter representations. However, the application of this division to the current form of the Wittgenstein ontology project is highly problematic: both classes and properties defined in the project documentation[28] are in fact representations of the structure of its subject-matter; that is, the Wittgenstein *Nachlass* and Wittgenstein’s publications. Nevertheless, it is useful to separate the “hard” or “general” part of the knowledge base from its numerous individual components that throng in separate classes.

The “Subject” branch of the project is much more suited for the division in question: we need to create a mechanism or structure of representation that would be applied more universally than just for Wittgenstein’s thought. Therefore, the main

[28] See P i c h l e r and G j e s d a l, *Wittgenstein Ontology.*

"Subject" subclasses: "Perspective," "Claim," "Concept," or "Ruleset" do not say anything specific about the philosophy of the author of the *Tractatus*. Similarly, some of the object and data properties that would figure in the completed knowledge base would belong to the universal structure that would allow us to grasp the variety of knowledge in the humanities. These would be the meta-level resources. However, other properties would be specific to this particular task: they would be introduced to represent aspects of some of Wittgenstein's ideas. Therefore, they would be subject-mater level resources.

Secondly, we should be aware that each attempt at representing philosophical content by means of the SW technology would be a form of translation: it would boil down to casting the natural language of a given philosophical work onto a much less expressive semiformal system whose set of semantic elements is strictly controlled. Therefore, a knowledge base developer would have to make many interpretative decisions not only to narrow down the number of separate lexical units but also to figure out which aspect of the meaning of a given word is at play in a given phrase.

The verb "is" usually raises particularly difficult interpretative questions. If, for instance, the *Tractatus* 6 states: "The general form of a truth-function *is* $[\bar{p}, \bar{\xi}, N(\bar{\xi})]$. This *is* the general form of a proposition," we are inclined to say that the "is" in the second sentence expresses the synonymity of the two complex concepts. However, the "is" in the first sentence does something else: here, the concept is juxtaposed with a complex symbol (or rather, to use Tractarian terminology, a complex sign), so we should instead read "is" as "is expressed with." In any case, this is yet another reason why we need to include perspectives in our attempts to build representations of philosophical conceptions: each such decision can be undermined by other scholars, and they should also have a possibility to present their own visions.

Having made these preparatory remarks, we can now turn to the main issue: how the three "Subject" subclasses: "Perspective," "Claim," and "Concept" can be employed to the task of representing knowledge in the humanities, or, more specifically, in the philosophy of Wittgenstein. Let us begin the description from a "Perspective" class member. A perspective can be considered to be a set of beliefs, and this should be the role of an entity of that class: it should have a certain number of properties whose values would be entities representing claims. We can label this kind of property simply as "claim"; it should have its domain in "Perspective" and range in "Claim" (the class).

A value of the "claim" property (that is, a member of the "Claim" class) would be an entity with two crucial properties. We would call them "source of claim" and "structure of claim"; the value of the former would be a member of the "Sentence" class from the "Source" branch, which is how we can anchor the subject-matter in its material, textual ground. However, the most important part of this construction would be the value of the second property: it would be an entity of the built-in type "Statement." In other words, the structure of the claim would be modelled thanks to the RDF reification syntax. Diagram 12 illustrates this strategy with a complex graph for a relatively simple example: the first sentence of the *Tractatus* 6.

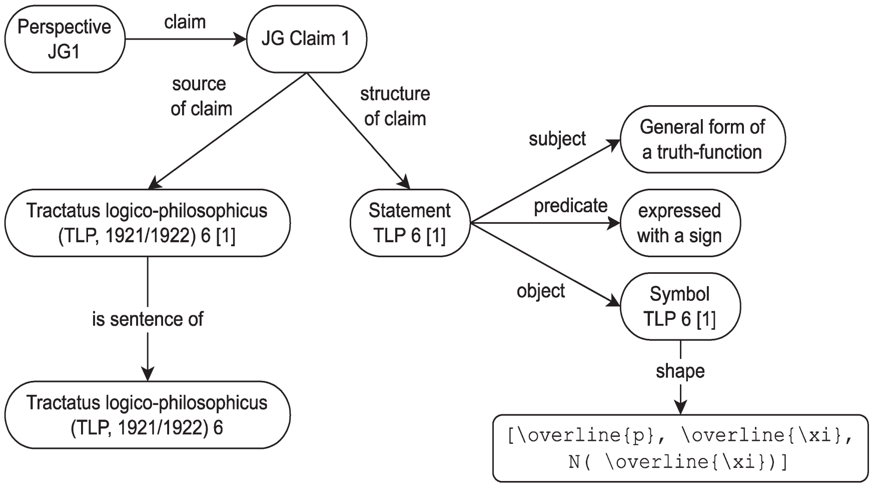
Perspective JG1
claim
JG Claim 1
source of claim
structure of claim
subject
General form of a truth-function
Tractatus logico-philosophicus (TLP, 1921/1922) 6 [1]
Statement TLP 6 [1]
predicate
expressed with a sign
object
Symbol TLP 6 [1]
is sentence of
shape
Tractatus logico-philosophicus (TLP, 1921/1922) 6
[\overline{p}, \overline{\xi}, N(\overline{\xi})]

Diagram 12. The pattern of representing perspectival knowledge.

The entity in the bottom left corner of the graph is an existing member of the "Part" class that appears in the current version of the Wittgenstein ontology OWL file. All the rest is not yet present there. As can be guessed, the node "Symbol TLP 6 [1]" is meant to be a member of the "Symbol" class, and is meant to have its property "shape" filled with a TEX-style string. The element "expressed with a sign" is an object property that stands for "is" in the represented sentence as we have briefly discussed above. The definitions of the resources that figure as values of the three reification properties "subject," "predicate," and "object" can be given outside of the context of the perspective, although there is a reason why we should think twice about that. After all, such extra-reification definitions would repeat the old foundationalist epistemology pattern, according to which there is a level of unproblematic simple building blocks, and all meaningful disputes regarding the structures of representations require agreement about the entities occurring at that zero-level.[29]

The first sentence of the *Tractatus* 6 is a relatively simple case of a structure that can be represented by a triple. What about more challenging examples? One of them is illustrated by the graph in Diagram 13. Here we have a possible structure of the first sentence of the *Tractatus* 6.2322 that says: "It is impossible to assert the identity of meaning of two expressions." As we can see, the right side of the graph differs significantly from the previous example: we have a nested reification because the value of the "subject" property of the external statement is also a statement (and a blank node).

[29] In other words, we would fall into the trap of the Sellarsian "myth of the given" (see Wilfrid S e l l a r s, *Empiricism and the Philosophy of Mind*, ed. Robert B. Brandom (Cambridge: Harvard University Press, 1997)).

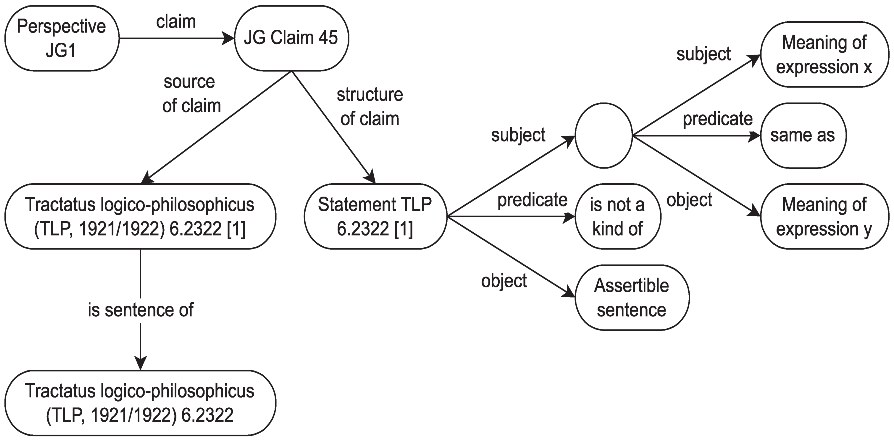

Diagram 13. Another, more complicated version of the pattern from Diagram 12.

This is only one possible variation on the theme: the starting structure from Diagram 12. We can also, for instance, multiplicate the "structure of claim" property and thus attribute two or more base reifications to one "Claim" entity. This should be the case for the first sentence of the *Tractatus* 6.22 that served us in the first section to illustrate the idea of a blank node (see Diagram 7).[30] Thus, we can make a representation of any given syntactic structure.

The application of the reification syntax has one major advantage: our knowledge base can still remain consistent despite including two (or more) claims that explicitly contradict each other because reified statements are not asserted. Moreover, in principle there is nothing that prevents us from modelling inconsistent perspectives. However, this comes at a certain price: firstly, the standard way to generate new information is blocked; secondly, queries become much more complicated as well as much less clear; thirdly, the standard SW development tools would not help in modelling such a knowledge base. We need to address the three problems now.

Let us begin with the last one. OWL files are usually not written manually in code editors: there is an application that provides a convenient graphical interface and automates some processes related to the development of a computational ontology. The application, called Protégé, was created by the Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine. It is open-source software that can be downloaded for free from its website[31] or used as a web

[30] It should be evident that the graph presented in Diagram 7 cannot simply replace some part of Diagram 12. Moreover, all the blank nodes used in the reifications must be named because they usually occur in more than one reified triple.

[31] See Protégé, https://protege.stanford.edu.

application. It provides a significant facilitation, especially when one is dealing with a complicated knowledge representation containing a large number of entities.

The problem is that Protégé does not support the RDF reification mechanism. Therefore, in order to be capable of developing a perspectival representation of philosophical content, one needs a different tool that would provide a graphical interface suitable for the extensive use of reifications. It should be noted that a comprehensive graphic representation of a knowledge base that includes reifications is not easy to design. There should be a kind of "surfacing" mechanism that would allow for the comprehensive inspection of information represented in the reifications related to one and the same perspective.[32]

Addressing the uncertainties regarding the structure of queries over a reification-pervaded knowledge base, we can offer a similar solution: we simply need a new piece of software that would translate between complicated RDF representations and more user-friendly constructions. The software would work as an overlay on the SPARQL interface: it would take a certain form of perspective-oriented query language and produce proper SPARQL expressions that would be sent to a knowledge base constructed according to the design pattern proposed here. Let us observe that such an overlay would not affect SPARQL's internal syntax, nor would it require any change in the query engine of a knowledge base.

The last thing we must address here is the system's inability to generate new knowledge from knowledge represented via the reification structure. This is a rather obvious ramification of the fact that a reified triple lacks assertion. The same fact enables us to store contradictory representations in our base without making it incoherent; therefore, we should rather think of disabling the reasoning mechanism as one of the features of the proposed solution. In fact, the disabled mechanism providing inferences for non-reified chunks of information operates according to the rules of first order logic: only a few things in the humanities can be done in this way. Instead, we should think of different kinds of reasonings in the humanities and their possible implementation in the system.

Non-FOL automated reasoning is a very broad issue that can be addressed in this paper only fragmentarily. At this point, we arrive at the application of the "Ruleset" class. As has been mentioned above, it is a part of a mechanism that utilises elements of the represented content to build inference rules for generating new information. First of all, what we demonstrate here is only one of many potential ways of using that class. The way we will discuss this utilises "Symbol" entities, but much more interesting applications of the mechanism would involve concepts.

[32] This would mean that the proposed application must be developed specifically for the solution outlined in Diagram 12. However, the solution is not a single-purpose mechanism; it can be applied to a broad spectrum of knowledge in the humanities.

A “Ruleset” exponent would have a number of meta-level properties. One such property would be “recursive definition.” The value of that property would be a structure made of three blank nodes. The main blank node, the direct value of the property, would be of an RDF built-in type Seq (not discussed in the section devoted to the elements of the SW technology); in other words, it would represent a sequence. In this case, the sequence would have two members given via the special RDF properties “_1” and “_2.” The other two blank nodes would be values of those properties. Their types would be left indefinite, but each of them would have two meta-level data properties: “base” and “derived.” Those properties’ values would be TEX-formatted strings. Diagram 14 illustrates the appropriate named graph.

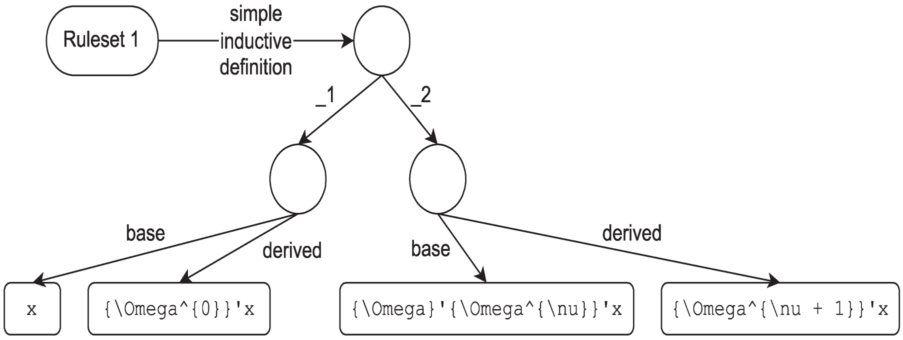

Diagram 14. A ruleset example.

As we can conjecture from the contents of the TEX-strings in the graph, the whole mechanism can be used to store recursive definitions such as the first definition in the *Tractatus* 6.02. Firstly, we provide the rule for the first member of a series (the value of the property “_1”); next, we give the rule for a general member (the value of the property “_2”). We complicate the representation of that piece of the *Tractatus* so much in order to be able to use it to automatically generate a result for any given argument. According to the first definition in 6.02, for the argument x we should get the result Ω^{0}’x; for Ω’x, we should get Ω^{0+1}’x; for Ω’Ω’x, we should get Ω^{0+1+1}’x; and so on.

Let us explain that once more, this time with the help of the Turtle notation:

```
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
prefix ex: <http://example.com#> .

ex:Ruleset1 rdf:type ex:Ruleset .
ex:Ruleset1 ex:simple_recursive_definition [
rdf:_1 [ ex:base “x”; ex:derived “{\Omega^{0}}’x” ];
rdf:_2 [ ex:base “{\Omega}’{\Omega^{\nu}}’x”;
      ex:derived “{\Omega^{\nu + 1}}’x” ] ] .
```

As we enter the above code in our hypothetical knowledge base, we would write the SPARQL query as follows:

```
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
prefix ex: <http://example.com#> .

SELECT ?d WHERE { ex:Ruleset1 ex:simple_recursive_definition ?aux1 .
            ?aux2 ex:base "{\Omega}'{\Omega}'{\Omega}'{\Omega'}'x" .
            ?aux2 ex:derived ?d .
            ?aux1 ex:derive ?aux2 . }
```

There are four RDF triples in the WHERE-clause of the query. The first of them asks for the value of the "simple recursive definition" property that is attributed to the "Ruleset1" entity. This value is then stored in the variable "aux1." The next triple defines an anonymous node (stored in the variable "aux2") that has the property "base" with a value "{\Omega}'{\Omega}'{\Omega}'{\Omega'}'x;" this string is our argument. The third triple links the anonymous node "aux2" with the output variable "d" by the property "derived," so the named graph for "aux2" conforms to the named graph of the blank node that is the value of the RDF "_2" property.

The real magic happens in the last triple: here, we have the property "derive" that is not given as a property of the main blank node in the definition of "Ruleset1." This is to be one of the magic properties in the mechanism of deriving anything with the help of "Ruleset" entities. In other words, some apparent properties must be hidden meta-keywords written in the SPARQL query engine running on the knowledge base server. As the query engine encounters one of them in the course of processing a query, it launches an appropriate script that computes a desired output value based on data that has already been retrieved. In our example, the output value would be "{\Omega^{0+1+1+1+1}}'x," and the script that would compute recursive definitions operating on symbols' syntax is relatively simple to write.

The main charge against solutions of that kind would certainly point to the fact that in order to implement them, we need to make modifications to the SPARQL interpreter. Unlike in the case of the RDF reification mechanism, where one only needs a new tool, here we attempt to make changes to one of the core SW technologies. This can raise concerns regarding compatibility. However, it can be noted that the proposed changes do not affect any existing SPARQL mechanisms; they only complement them with some additional capabilities. This means that the backward compatibility will be preserved.

Again, the complexity of the actual query calls for some sort of overlay software that would translate more user-friendly commands into the modified SPARQL code.

*

It is no secret that the SW paradigm is suited to a different kind of knowledge than what one deals with in the humanities. The WAB exponents use the juxtaposition of "jigsaw puzzle" and "crisscross" knowledge: in the most SW-friendly cases, all the pieces of the represented domain have their well-defined places[33]. Philosophy is very far from being such a domain: its categories are dynamic, overlapping, context-sensitive, and open to various interpretations; there is no universal agreement about virtually anything in philosophy, save maybe such simple facts as the number of the *Tractatus*' theses. Perhaps philosophy is the least SW-friendly case.

Nevertheless, in the present paper we have discussed the prospects of a project that aims at providing an SW knowledge base that would cover the philosophy of Wittgenstein. Over the past fifteen years, a group of WAB-linked scholars has made attempts at figuring out how one can utilise RDF and OWL to produce the most adequate representation of "crisscross" knowledge. Their theoretical achievements are notable: they have come up with the idea of an ontology that allows for internal contradictions thanks to the introduction of individual perspectives. According to the WAB team, perspectives meant to embrace non-contradictory chunks of information related to the same set of concepts[34]. The present paper attempts to flesh out the idea with the help of the RDF reification mechanism. The solution appears to be even more liberal than the theory: a single perspective can include contradictory claims due to the fact that reified RDF triples are not taken as arguments by automated reasoners; in other words, reification strips assertion from statements.

As has been noted in the preceding section of the paper, having the automated inferential mechanism disabled is not a bad thing for a knowledge base in the humanities; we should instead think of different mechanisms of reasoning that are more suitable to the subject matter. In the present paper, one such mechanism was presented: it was an idea of a reasoner that would work on the basis of a recursive definition whose specification would be dependant on the actual content of the knowledge base (in this case, the content was the first definition from the *Tractatus* 6.02).

However, the same mechanism can be applied in a very different way: we can create reasoners that utilise statistical computations and advance pattern recognition algorithms based on neural networks. We can build systems of inferences that learn to provide satisfactory answers to problems that have no definite solutions. We can apply AI deep learning algorithms. All these possibilities will be available under the

[33] See P i c h l e r, F i e l d i n g, G a n g o p a d h y a y, and O p d a h l, "Crisscross Ontology: Mapping Concept Dynamics, Competing Argument and Multiperspectival Knowledge in Philosophy," 59–60.

[34] See ibidem, 69; M á c h a, F a l c h, and P i c h l e r, "Overlapping and Competing Ontologies in Digital Humanities."

condition that we decide to modify the SPAQRL interpreter to let it recognise "magic" properties that can launch external scripts.

One can notice that with advanced statistical reasonings and neural networks, we are no longer dealing with SW systems in a classic sense. Firstly, it would instead be a mixed system that would combine the elements of the GOFAI with the artificial neural network paradigm. It is worth noting that such a combination is broadly recognised as the future of all AI technology.[35]

Secondly, a reasoner would no longer provide fully consistent and precise knowledge. Moreover, it would be possible for two identical reasoners to give different answers based on a common knowledge base; furthermore, one reasoner would give different answers each time it was queried. Therefore, if we enable a reasoner to add its answers to its knowledge base every time it is queried, the knowledge base in question would develop in a unique way. One can consider such a possibility as a potential flaw. However, the idea of a software agent that meanders in a not entirely predictable way fits the philosophy much better than a rigid FOL inference system. In other words, we should consider this to be a rather promising feature.[36]

BIBLIOGRAPHY / BIBLIOGRAFIA

Allemang, Dean, Jim Hendler, and Fabien Gandon. *Semantic Web for the Working Ontologist. Effective Modeling for Linked Data, RDFS, and OWL*. New York: Association for Computing Machinery, 2020.

Brickley, Dan, and R.V. Guha. *RDF Schema 1.1*. W3C. February 25, 2014. https://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

Gruber, Thomas R. "Translation Approach to Portable Ontology Specifications." *Knowledge Acquisition* 5 no. 2 (1993): 199–220.

Kautz, Henry. "The Third AI Summer: AAAI Robert S. Engelmore Memorial Lecture." *AI Magazine*, no. 43 (2022): 105–25.

[35] See Henry K a u t z, "The Third AI Summer: AAAI Robert S. Engelmore Memorial Lecture," *AI Magazine*, no. 43 (2022): 118–20. Kautz notices that there can be a couple of different forms of the merge of the two approaches. According to him, the most promising approach for merging the approaches is to embed the symbolic (GOFAI) reasoning engine as a subroutine of the neural engine (see ibidem, 119). The approach that would result from the proposal of extending SPARQL interpreter goes in the opposite direction: the neural engine is a subroutine of the symbolic reasoner.

[36] I would like to thank the scholars who helped me with this paper. My thanks are due, in particular, to Alois Pichler, Matthew Fielding, Andreas Opdahl, Rune Falch, Wilhelm Krüger, and Grzegorz Nalepa; all of them were ready to answer my numerous questions regarding Semantic Web technology and the Wittgenstein ontology project. I am indebted also to Filip Mazurczak and the two anonymous reviewers for their useful comments and linguistic corrections on an earlier version of this manuscript.

Mácha, Jakub, Rune Falch, and Alois Pichler. "Overlapping and Competing Ontologies in Digital Humanities." In *DH-CASE '13: Proceedings of the 1st International Workshop on Collaborative Annotations in Shared Environment: metadata, vocabularies and techniques in the Digital Humanities (10 September 2013, Florence, Italy)*. Edited by Francesca Tomasi and Fabio Vitali. New York: Association for Computing Machinery, 2013.

Marshall, Catherine C., and Frank M. Shipman. "Which semantic web?" *HYPERTEXT'03: Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia*, (2003): 57–66.

Pichler, Alois. *Wittgenstein Ontology*. Bergen: University of Bergen, 2013. http://wab.uib.no/wab_philospace.page.

Pichler, Alois, and Amélie Zöllner-Weber. "Sharing and Debating Wittgenstein by Using an Ontology." *Literary and Linguistic Computing* 28, no. 4 (2013): 700–7.

Pichler, Alois, James Matthew Fielding, Nivedita Gangopadhyay, and Andreas Lothe Opdahl. "Crisscross Ontology: Mapping Concept Dynamics, Competing Argument and Multiperspectival Knowledge in Philosophy." *Quaderni di Filosofia*, no. 2 (2021): 59–73.

Pichler, Alois, and Øyvind Liland Gjesdal. *Wittgenstein Ontology*. Bergen: University of Bergen, 2007. http://ubbdev.gitlab.io/wab-ontology/index-en.html.

Prud'hommeaux, Eric, and Andy Seaborne. *SPARQL Query Language for RDF*. W3C. January 15, 2008. https://www.w3.org/TR/rdf-sparql-query/.

RDF Working Group. *Resource Description Framework (RDF)*. W3C. February 25, 2014. https://www.w3.org/RDF/.

Sánchez, Diana Marcela, José María Cavero, and Esperanza Marcos Martínez. "The Road Toward Ontologies." In *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems*. Edited by Raj Sharman, Rajiv Kishore, and Ram Ramesh. New York: Springer, 2007.

Sellars, Wilfrid. *Empiricism and the Philosophy of Mind*. Edited by Robert B. Brandom. Cambridge: Harvard University Press, 1997.

W3C OWL Working Group. *OWL 2 Web Ontology Language*. W3C. December 11, 2012. https://www.w3.org/TR/2012/REC-owl2-overview-20121211/.

Wittgenstein, Ludwig. *Tractatus Logico-Philosophicus*. Translated by David F. Pears and Brian F. McGuinness. London–New York: Routledge and Kegan Paul, 1965.

Zöllner-Weber, Amélie, and Alois Pichler. "Utilizing OWL for Wittgenstein's Tractatus." In *Papers of the 30th International Ludwig Wittgenstein Symposium (5–11 August 2007, Kirchberg am Wechsel)*. Edited by Herbert Hrachovec, Alois Pichler, and John Wang. Kirchberg am Wechsel: ALWS, 2007.

ABSTRACT / ABSTRAKT

Jakub GOMUŁKA, AI Applied to Philosophy: A Contribution to the Wittgenstein Ontology Project

DOI 10.12887/36-2023-3-143-12

The paper focuses on the Wittgenstein Ontology Project run by the Wittgenstein Archives at the University of Bergen (WAB). The project is an attempt to apply Semantic Web (SW) technology to the task of making Wittgenstein's philosophy available as a searchable knowledge base on the Internet. The SW is one of the paradigms of Knowledge Representation and Reasoning (KR^2) research and is a descendant of the symbolic AI research programme (also known as 'Good Old-Fashioned AI' or GOFAI). After a brief introduction to the SW technology, the paper discusses the main difficulties in applying it to the humanities and to the philosophy of Wittgenstein in particular. Next, it turns to the WAB team's attempts to deal with these difficulties, especially a much discussed perspectival approach that would allow for a representation of competing or contradictory claims. Finally, some original solutions are proposed: namely, the implementation of the perspectival approach with the help of the RDF reification mechanism and a new mechanism of reasoning that allows rules of inference to be generated from the content of a knowledge base. The latter solution requires a significant change in present-day technologies; that is, in the SPARQL interpreter. However, it is also a rather promising extension of the present paradigm that can potentially allow the merging of GOFAI with the artificial neural network approach.

Keywords: Ludwig Wittgenstein, Semantic Web technology, computational ontology, symbolic artificial intelligence (AI)

Contact: Faculty of Humanities, AGH University of Science and Technology, ul. Czarnowiejska 36, 30-054 Cracow, Poland
E-mail: jgomulka@agh.edu.pl
Phone: +48 12 6174365

Jakub GOMUŁKA – Sztuczna inteligencja zastosowana do filozofii. Wkład w rozwój projektu Wittgenstein Ontology

DOI 10.12887/36-2023-3-143-12

Artykuł dotyczy projektu Wittgenstein Ontology realizowanego przez Archiwa Wittgensteina przy Uniwersytecie w Bergen (WAB). Projekt ten jest próbą zastosowania technologii Sieci Semantycznej (SW) do udostępnienia filozofii Wittgensteina w internecie w postaci przeszukiwalnej bazy wiedzy. SW jest jednym z paradygmatów badawczych w ramach nurtu w informatyce zwanego Knowledge Representation & Reasoning (reprezentacja wiedzy i wnioskowanie – w skrócie KR^2), będącego spadkobiercą programu badawczego symbolicznej sztucznej inteligencji (określanego również starą dobrą sztuczną inteligencją). Po krótkim wprowadzeniu do technologii SW w artykule omówiono główne trudności w stosowaniu jej do humanistyki, a w szczególności do filozofii Wittgensteina. Następnie pokazano, w jaki sposób zespół WAB próbuje przezwyciężyć te trudności, przy czym uwagę skierowano głównie na obszernie omawiane przez członków tego zespołu podejście perspektywiczne, które pozwalałoby reprezentować rywalizujące ze sobą czy wręcz sprzeczne tezy. W głównej części artykułu zaprezentowano parę oryginalnych rozwiązań, to znaczy implementację podejścia perspektywicznego przy użyciu mechanizmu reifikacji wbudowanego w składnię RDF, a także nowy mechanizm rozumowania pozwalający tworzyć reguły inferencji na podstawie treści reprezentowanych w danej bazie wiedzy. To ostatnie rozwiązanie wymaga poważnej zmiany w aktualnie używanych technologiach, ściślej mówiąc, w interpreterze SPARQL. Jest ono jednak bardzo obiecującym rozszerzeniem obecnego paradygmatu, które pozwala na połączenie w pracach nad sztuczną inteligencją tradycyjnego podejścia symbolicznego z podejściem wykorzystującym sztuczne sieci neuronalne.

Słowa kluczowe: Ludwig Wittgenstein, technologia Sieci Semantycznej, ontologia komputacyjna, symboliczna sztuczna inteligencja

Kontakt: Katedra Technologii Informacyjnych i Mediów, Wydział Humanistyczny, Akademia Górniczo-Hutnicza im. Stanisława Staszica, ul. Czarnowiejska 36, 30-054 Kraków
E-mail: jgomulka@agh.edu.pl
Tel. 12 6174365