Cite as: Bergman, M. K. KR Vocabulary and Languages. in *A Knowledge Representation Practionary: Guidelines Based on Charles Sanders Peirce* (ed. Bergman, M. K.) 151–180 (Springer International Publishing, 2018). doi:10.1007/978-3-319-98092-8_8

Official site: https://link.springer.com/book/10.1007/978-3-319-98092-8

Full-text: http://www.mkbergman.com/publications/akrp/chapter-8.pdf

Abstract: We use deductive reasoning to infer hierarchical relationships, create forward and backward chains, check if domains and ranges are consistent for assertions, assemble attributes applicable to classes based on member attributes, conform with transitivity and cardinality assertions, and check virtually all statements of fact within a knowledge base. As inductive and abductive reasoners become available, they will expand this list of capabilities to include question answering, hypothesis generation and testing, forecasting, decision-making, and real-time systems in robotics. We want a knowledge representation (KR) language that can model and capture intensional and extensional relations; one that potentially embraces all three kinds of inferential logic; one that is decidable; one that is compatible with a design reflective of particulars and generals; and one that is open world in keeping with the nature of knowledge. Our choices for the knowledge graph is the W3C standard of OWL 2 (the Web Ontology Language) and for data representation is RDF (Resource Description Framework).

8

KR VOCABULARY AND LANGUAGES

We have now armed ourselves with basic terminology and a framework around which to express a starting vocabulary for knowledge representation. However, we have one question to answer before we can adopt the languages (or symbol systems) we need to convey that vocabulary: What current languages can capture Peirce's theory of logic while being consistent, coherent, and practical for our needs in knowledge representation? To resolve that question, we need to delve deeper into Peirce's logic and options provided by current language choices. The practical choices resulting from these intersecting forces will then enable us to specify our starting KR vocabulary into a working language suitable for computers.

A vocabulary, in the sense of knowledge systems or ontologies, may and should be expandable, but it is also a controlled vocabulary.[1] That is, we declare new terms and relations to the system and define them at levels required by the formalism to achieve the vocabulary's purpose. Terminology is a social process, driven by user needs and the occasional 'surprising fact,' such as the emergence of the Internet or smartphones, that requires our terms to adapt and our knowledge to grow. Our vocabularies also serve other purposes, such as providing consistent labels to user interfaces or helping interoperate with other knowledge sources.

Early in my exposure to semantic technologies, I encountered the phrase 'ontological commitment.'[*] This phrase was common in the early literature, and, for some reason, I found the idea off-putting. It seemed to me it conveyed buying into one ontology versus another, and I did not like the idea of boxing myself in. There is a significant plurality within the semantic technology community that does not like the idea of a governing schema. what is sometimes pejoratively called "one ring to rule them all."[3]

I have come to embrace a very different view when it comes to the idea of representing knowledge representation. If one believes in reality and truth, and that the purpose of knowledge is to further the understanding of truth, then a knowledge representation system must be based on logics and formalisms that can capture knowledge of every sort and can provide coherent and testable means for discovering and testing new knowledge.

* See, for example, *Davis et al.*[2] from 1993.

As we will see in the context of the logics and the languages we have chosen for this task, the idea is not to decide what is true and what is false. Our primary objective is not to pass judgment on such topics as fake news. Instead, the design is to test new information we introduce to the system — a system which we deem already to be correct and true within the limits of our assertions — as to whether that new information is consistent and coherent. We first test for consistency in syntax and grammar, and for minimally acceptable completeness. If the new information meets the threshold and does not violate the knowledge graph already in place — that is, we deem it coherent — we accept that new information. If the new information meets the threshold but violates what we already have, we either reject it as incoherent, or we revise the existing assertions to reflect this new information. In this way, new information may cause us to change the knowledge graph in the system. True, we could extend this basis to test for the falsity of news or other information. However, the more technical point is that we premise our starting basis on 'open' assumptions consistent with the nature of knowledge, as I discuss in the next chapter.

These absolute groundings are further essential to provide a consistent and logical basis for computers to test and analyze current and new assertions. We want the representations available (that is, *features*) for machine learning to reflect and adapt to truth as we test it. While I dislike the phrase 'ontological commitment,' its very scariness helps cement the importance of inspecting (and *re*-inspecting) the formalisms upon which we base our knowledge representation systems.

In creating a feature-rich logic machine for AI, we, of course, want a system that is *sound*, *consistent*, *coherent*, and relatively *complete*. *Sound* is an evaluative criterion where all provable statements are valid in all models. *Consistent* is where all axioms* in the knowledge base (domain), subject to deductive reasoning, are true (or, at least, exhibit no contradictions). *Coherent* is where the knowledge base (domain) is consistent and has a high degree of conjunction for non-deductive assertions.† C*omplete* is an evaluative criterion where all statements that are true in the model are provable and meet minimum standards.

Our interpreters are both artificial agents and humans. We need us, as humans, to scope the domain and provide the vocabulary, then to construct and oversee the knowledge graph, and then to maintain and extend the system, and lastly to review tests and tentative assignments before we agree to commit to the knowledge base. These aspects must be suitable for direct translation into any human language. Humans will also have many non-AI uses for the knowledge system, reinforcing the need for understandability and usability. To complete the speculative grammar leg of Peirce's theory of logic, we also need to capture factors relevant to the Secondness of *critic*, the methods of logic, and to the Thirdness of the *methodeutic*, the application of these methods to practical problems addressed with practical means.

* An *axiom* is a premise or starting point of reasoning. In an *ontology*, each *statement* (assertion) is an axiom.

† *Coherence* has a long history within epistemic logic,[4]with more recent trends toward accommodating probabilistic measures, though specific methods lacking broad consensus. Peirce's views would tend to align coherence with *abduction* and *pragmatism*. Researchers that embrace aspects of Peirce's approach include Roche[5] and Douvan and Meijs.[6]

LOGICAL CONSIDERATIONS

KR practitioners know that the choice of a formalism (*i.e.*, syntax and language) for knowledge representation involves a trade-off in expressivity and practicality.[7] Knowledge graphs and knowledge bases should be scoped based on their anticipated domain of use and populated with 'vivid' knowledge.[8] We cannot feasibly specify all aspects of all items while remaining computationally tractable. We need to *infer*[*] connections, properties, and relationships without explicitly stating all of them. When we do make assignments, we need to know what those statements *entail* based on prior statements.[†] We base inference and entailment on the underlying KR language, as applied through its defined and understood vocabulary and semantics.[9]

We could derive the logic that governs our semantics from multiple logic families. One option is propositional logic, but unfortunately this logic evaluates statements such as "Aristotle is a philosopher" and "Plato is a philosopher" as unrelated. Another option is set theory, with its basis in sets and members and strong mathematical background. However, set theory lacks predication and ideas of identity and can be problematic for the largest of sets. Advances have continued in set theory, such that the basis for a complete KR language may be at hand.[10] But these limitations led mathematicians and logicians in the 19th century, noted previously, to work out the new logic of relations. We now call this field predicate calculus or first-order logic (FOL). Paternal rights to FOL are granted to both Peirce[7]and Frege, though both apparently worked without the others knowledge.[12]

Peirce attempted to explicate the basis, applicability and interpretation of deductive, inductive, and abductive logic, the latter of which he introduced to modern logic. Peirce's primacy of logic proceeds as follows: Decompose every statement into its fundamental premises. Conduct all logical tests, including the implications resulting from inference. Single out anomalies or 'surprising facts' for special attention subject to the pragmatic maxim. Every bit of Peirce's logical work has applicability to knowledge representation.

First-order Logic and Inferencing

First-order logic is superior to propositional logic in that it allows variables and quantifiers. FOL enables us to say 'x is a philosopher,' where we treat the subject as a variable and turn 'is a philosopher' into a predicate. We can establish relations between these predicates with logical connectors, such as AND, OR, NOT and IF-THEN statements. We may base class membership on *intensional*[13] or *extensional*[14] grounds. We can apply quantifiers to statements using universal ('for every') and existential

* *Inference* is the act or process of deriving logical conclusions from premises known or assumed as true. The logic within and between *statements* in an *ontology* is the basis for inferring new conclusions from it, using software applications known as inference engines or *reasoners*.

† *Entailment* is a consequence arising from a statement deemed true based on some underlying logic. The logical consequence is said to be *necessary* and *formal*; necessary, because of the rules of the logic (the conclusion is the consequent of the premises); and formal because the logical form of the statements and arguments hold true without regard to the specific *instance* or content.

('there exists') quantifiers, as well as use negation. FOL is *sound* (all provable statements are true in all models) and *complete* (all statements which are true in all models are provable). Moreover, Peirce based his 'beta' version of existential graphs on FOL plus identity. Because of these efforts, we view Peirce as one of the founders of first-order logic.

FOL is a powerful and expressive formalism for knowledge representation.[15] Gödel's completeness theorem proved that deduction in FOL is sound and complete. Still, Alonzo Church and Alan Turing proved independently, in 1936 and 1937, that FOL under certain conditions, notably the halting problem or quantification over infinite sets, was undecidable. What this means is a computer may never calculate some problems to a final result. Various options that reduce the expressivity of FOL have been formulated to overcome the problem of reliably computing to completion. We will speak of one of them, descriptive logics, shortly.

One of the KR formalisms, conceptual graphs, is a complete expression of FOL and is patterned on Peirce's existential graphs.[16] Some proponents call for KR formalisms that can handle higher-order logics, such as predicates of predicates. Recent attempts to bridge description logics to category theory using underlying ideas are intriguing and redolent of Peirce.[17] KR and its logical and formal underpinnings, I believe, are set to undergo a renaissance.

Inferencing is the drawing of new facts, probabilities or conclusions based on reasoning over existing evidence. Inferencing is a common term heard in association with semantic technologies. Inference engines (also known as *reasoners*, semantic reasoners, reasoning engines, or rules engines) are the application components. Peirce classed inferencing into three modes: deductive reasoning, inductive reasoning, and abductive reasoning. Deductive reasoning extends from premises known as true and clear to infer new facts. Inductive reasoning looks at the preponderance of evidence to infer what is probably true. Abductive reasoning poses possible explanations or hypotheses based on available evidence, often winnowing through the possibilities based on the total weight of evidence at hand or what is the most practical explanation. Though we may apply all three reasoning modes to knowledge graphs, the standard and most used form is deductive reasoning. Knowledge base completion, a new field, sometimes uses inductive reasoning. Abductive reasoning, to my knowledge, has not been applied to knowledge graphs. Inductive and abductive logics offer much additional leverage for a knowledge system, and I expect to see their use increase given their potential usefulness.

We can use inferencing to broaden and contextualize search, retrieval, and analysis. We can create inference tables in advance and layer them over existing data stores for speedier use and the automatic invoking of inferencing. More complicated inferencing means that models can also perform as complete conceptual views of the world or knowledge bases. Quite complicated systems are emerging in such areas as common sense and biological systems, as two examples.

We can apply inference engines at the time of graph building or extension to test the consistency and logic of the new additions. Additionally, we may apply semantic reasoners to a current graph to expand queries for semantic search or other reasoning purposes. As noted, as inductive and abductive reasoners become available, they will expand this list of capabilities to include question answering, hypothesis generation and testing, forecasting, decision making, and real-time systems in robotics. The contributions these types of tools will make is dependent upon the method of logic inference. Based on their syllogistic form, here is a comparison of the three methods:[18]

Deduction	*Induction*	*Hypothesis/ Abduction*[*]
All M is P (Rule)	S is M (Case)	S is P (Result)
S is M (Case)	S is P (Result)	All M is P (Rule)
S is P (Result)	So, All M is P (Rule)	So, S is M (Case)

Table 8-1: Syllogistic Forms for Inference Methods

Peirce explicated deductive and inductive reasoning in the clearest of ways and corrected erroneous views of what constituted inductive reasoning. He most importantly recognized that, just as many problems are distributive in nature, so are many of the logical questions. For this, Peirce decomposed the basic syllogism of the Greek philosophers to articulate a third kind of inference, abductive reasoning. "Deduction proves that something *must* be, Induction shows that something *actually is* operative, Abduction merely suggests that something *may be.*" (1903, EP 2:216)

Deductive Logic

Deduction is the "tracing out the consequences that would ensue upon the truth or falsity of that hypothesis" (nd, MS [R] S64). "By Deduction, or mathematical reasoning, I mean any reasoning which will render its conclusion as certain as its Premisses, however certain these may be." (1911, MS [R] 856:2)

Deduction is the most common logic in knowledge representations in their current form. We use deductive logic to infer hierarchical relationships, create forward and backward chains, to check if *domains* and *ranges* are consistent for assertions, assemble attributes applicable to classes based on member attributes, conform with transitivity and cardinality assertions, and check virtually all statements of fact within a knowledge base. In backward chaining, we conduct the reasoning tests 'backward' from a current consequent or 'fact' to determine what antecedents can support that conclusion, based on the rules used to construct the graph. ("What reasons bring us to this fact?") In forward chaining the opposite occurs; namely, we state a goal or series of goals, and then existing facts (as rules) are checked to see which ones can lead to the goal ("A goal X may be possible because of?"). The reasoner iterates the process until the goal is reached or not; if reached, we may add

* In his later years Peirce revised his views about abduction; see *Chapter 15*.

new knowledge using heretofore unstated connections to the knowledge base. We base consistency tests solely on deductive logic. Either an asserted statement *satisfies* specifications, or it fails.

Like so much Peirce did, he continued to refine his understanding of things through inspection and categorization. Here is one of his later formulations for deduction:

> "A *Deduction* is an argument whose Interpretant represents that it belongs to a general class of possible arguments precisely analogous which are such that in the long run of experience the greater part of those whose premisses are true will have true conclusions. Deductions are either *Necessary* or *Probable*. Necessary Deductions are those which have nothing to do with any ratio of frequency, but profess (or their interpretants profess for them) that from true premisses they must invariably produce true conclusions. A Necessary Deduction is a method of producing Dicent Symbols by the study of a diagram. It is either *Corollarial* or *Theorematic*. A Corollarial Deduction is one which represents the conditions of the conclusion in a diagram and finds from the observation of this diagram, as it is, the truth of the conclusion. A Theorematic Deduction is one which, having represented the conditions of the conclusion in a diagram, performs an ingenious experiment upon the diagram, and by the observation of the diagram, so modified, ascertains the truth of the conclusion. Probable Deductions, or more accurately, Deductions of Probability, are Deductions whose Interpretants represent them to be concerned with ratios of frequency. They are either *Statistical Deductions* or *Probable Deductions Proper*. A Statistical Deduction is a Deduction whose Interpretant represents it to reason concerning ratios of frequency, but to reason concerning them with absolute certainty. A Probable Deduction proper is a Deduction whose Interpretant does not represent that its conclusion is certain, but that precisely analogous reasonings would from true premisses produce true conclusions in the majority of cases, in the long run of experience." (1903, EP 2:297-298; CP 2.267-268)

At present, most current knowledge approaches only use what Peirce calls the 'necessary' deduction. Peirce placed deductive reasoning in Secondness, though he did consider other placements early in his career.* The placement in Secondness, however, does make sense because it is the logic of actualness, and whether actual things conform to the premises asserted for them.

<u>Inductive Logic</u>

Induction is "any reasoning from a *sample* to the whole sampled" (1911, NEM 3:178), with the sample taken at random. Induction is the probabilistic form of Peirce's reasoning triad. Peirce placed the inductive form of reasoning in Thirdness, consistent with its nature of potential.

Peirce wrote much on induction. One succinct summary is that "Induction consists in starting from a theory, deducing from it predictions of phenomena, and observing those phenomena to see how nearly they agree with the theory." (1903, EP

* Staat[19] concurs that the placement of the three types of inferential logic into Firstness, Secondness and Thirdness is the order of abduction, deduction and induction, though, when considered in the order of inquiry, it is abduction, induction, deduction. This has been a matter of some confusion to scholars.

2:216) And, "... observe that neither Deduction nor Induction contributes the smallest positive item to the final conclusion of the inquiry. They render the indefinite definite; Deduction Explicates; Induction evaluates: that is all." (1908, CP 6.475) Some of his longer passages from his later career expound further on this nature:

> "The validity of Induction consists in the fact that it proceeds according to a method which though it may give provisional results that are incorrect will yet, if steadily pursued, eventually correct any such error. The two propositions that all Induction possesses this kind of validity, and that no Induction possesses any other kind that is more than a further determination of this kind, are both susceptible of demonstration by necessary reasoning." (1906, NEM 4:319)

And:

> "The true guarantee of the validity of induction is that it is a method of reaching conclusions which, if it be persisted in long enough, will assuredly correct any error concerning future experience into which it may temporarily lead us. This it will do not by virtue of any deductive necessity (since it never uses all the facts of experience, even of the past), but because it is manifestly adequate, with the aid of retroduction and of deductions from retroductive suggestions, to discovering any *regularity* there may be among experiences, while *utter irregularity is not surpassed in regularity by any other relation of parts to whole*, and is thus readily discovered by induction to exist where it does exist, and the amount of departure therefrom to be mathematically determinable from observation where it is imperfect." (1908, CP 2.769)

Consistent with the universal categories, Peirce also saw three subdivisions, or types, within inductive reasoning, with the first being *crude induction*:

> "The first and weakest kind of inductive reasoning is that which goes on the presumption that future experience as to the matter in hand will not be utterly at variance with all past experience. Example: 'No instance of a genuine power of clairvoyance has ever been established: So I presume there is no such thing.' I promise to call such reasoning *crude induction*.... Crude induction is the only kind of induction that is capable of inferring the truth of what, in logic, is termed a universal proposition." (1908, CP 2.756-7)

The second type of induction is the strongest of the three, what Peirce called *quantitative induction*:

> "This [type] investigates the interrogative suggestion of retroduction, 'What is the 'real probability' that an individual member of a certain experiential class, say the S's, will have a certain character, say that of being P?' This it does by first collecting, on scientific principles, a 'fair sample' of the S's, taking due account, in doing so, of the intention of using its proportion of members that possess the predesignate character of being P. This sample will contain none of those S's on which the retroduction was founded. The induction then presumes that the value of the proportion, among the S's of the sample, of those that are P, probably approximates, within a certain limit of approximation, to the value of the real probability in question. I propose to term such reasoning *Quantitative Induction.*" (1908, CP 2.758)

Lastly, the third type, intermediate between the prior two:

> “The remaining kind of induction, which I shall call *Qualitative Induction*, is of more general utility than either of the others, while it is intermediate between them, alike in respect to security and to the scientific value of its conclusions. In both these respects it is well separated from each of the other kinds. It consists of those inductions which are neither founded upon experience in one mass, as Crude Induction is, nor upon a collection of numerable instances of equal evidential values, but upon a stream of experience in which the relative evidential values of different parts of it have to be estimated according to our sense of the impressions they make upon us.” (1908, CP 2.759)

In the first type, *crude induction*, we may only only detect falsity if we persist the inference long enough. In the strongest second type, *quantitative induction*, the sample is a sub-collection of a population of units; its inductive strength arises from being able to apply the theory of errors. The third type, *qualitative induction*, does not have the advantage of definite populations, but, as we enlarge the sample, the inferential evidence gets stronger. (1904, EP 2:302)

Inductive logic is only at the beginning phases of application to knowledge systems, with a leading computational approach for general purposes being inductive logic programming (ILP).[20] Induction has been used for question answering and to expand search[21] and in areas like knowledge base completion, learning,[22] and schema induction.[23] These thrusts deserve more attention, particularly in light of the Peircean bases emphasized throughout this book. Most machine learning involving knowledge bases is a form of inductive reasoning.

Abductive Logic

One of Peirce’s signal achievements was to bring the idea of abduction to modern logic. Peirce wrote and revised his views on abduction* over his entire working life. A consistent thread in his characterization was that abduction is a kind of inference that originates a hypothesis by concluding in an explanation, though an indeterminant one for a given observation, often of a curious or surprising nature. Peirce studied abduction because of his belief in its essential role in the scientific method, as well as the unique inferential and logical possibilities it allowed. Peirce went so far as to state that pragmatism is the “logic of abduction” (1903, CP 1.595 *ff.*). He also called the combination of abduction with induction an ‘analogy’ (1896, CP 1.65). In 1903 he offered the following syllogistic form for abduction (CP 5.189, EP 2:231):

> “The surprising fact, *C*, is observed;
> But if *A* were true, *C* would be a matter of course,
> Hence, there is reason to suspect that *A* is true.”

This part of the inference chain begins with the ‘surprising fact’ or an event or question. By 1911, however, Peirce wrote, “I do not, at present, feel quite convinced that any logical form can be assigned that will cover all ‘Retroductions’ [abductions]. For what I mean by a Retroduction is simply a conjecture which arises in the mind." (NEM 3:203-4) However, he also claimed that abduction is the "only kind of reasoning that opens new ground." (NEM 3:206) Though the syllogistic form still works, Peirce came to believe there was a qualitative and 'guessing' or 'instinctual' aspect to some abductions, which we need in any case to subject to the pragmatic test. In that same year of 1911 he more broadly stated:

> "By Retroduction [abduction*] I mean that kind of reasoning by which, upon finding ourselves confronted by a state of things that, taken by itself, seems almost or quite incomprehensible, or extremely complicated if not very irregular, or at least surprising; we are led to suppose that perhaps there is, in fact, another definite state of things, because, though we do not perceive any unequivocal evidence of it, nor even of a part of it, (or independently of such evidence if it does exist,) we yet perceive that this supposed state of things would shed a light of reason upon that state of facts with which we are confronted, rendering it comprehensible, likely (if not certain,) or comparatively simple and natural." (1911, MS [R] 856:3-4)

One way to understand Peirce's insight on abduction, though unclear this was his actual method, is to split the idea of hypothesis generation and testing into two parts and re-think their roles. In abduction, the conscious and unconscious mind when faced with a choice rapidly screens and mentally evaluates possible explanations for possible outcomes to test. Multiple possible pathways may explain the diverse potential results. Since the actual testing of a hypothesis using inductive logic incurs time and expense, we try to weigh, in our minds, the potential importance of the hypothesis and its likelihood of results against the time and cost to generate them. A careful weighing in our mind of potentials and costs invokes other signals and perceptions, some perhaps unconscious, such that we may often express our selections as a 'guess.' As part of his belief in the continuity of nature, however, Peirce also noted how often guesses are correct compared to random likelihood.

> "Abduction and induction have, to be sure, this common feature, that both lead to the acceptance of a hypothesis because observed facts are such as would necessarily or probably result as consequences of that hypothesis. But for all that, they are the opposite poles of reason, the one the most ineffective, the other the most effective of arguments. The method of either is the very reverse of the other's. Abduction makes its start from the facts, without, at the outset, having any particular theory in view, though it is motived by the feeling that a theory is needed to explain the surprising facts. Induction makes its start from a hypothesis which seems to recommend itself, without at the outset having any particular facts in view, though it feels the need of facts to support the theory. Abduction seeks a theory. Induction seeks for facts. In abduction, the consideration of the facts suggests the hypothesis. In induction, the study of the hypothesis suggests the experiments which bring to light the very facts to which the hypothesis had pointed. The mode of suggestion by which, in abduction, the facts suggest the hypothesis is by resemblance, -- the resemblance of the facts to the consequences of the hypothesis. The mode of suggestion by which in induction the hypothesis suggests the facts is by contiguity, -- familiar knowledge that the con-

* Alternate terms used by Peirce for *abduction* included retroduction, hypothesis, and presumption.

ditions of the hypothesis can be realized in certain experimental ways." (1901, CP 7.218)

In succinct terms, and Peirce's definition (1908) for retroduction, it is "the passage of thought from experiencing something, E, to predicating a concept of the mind's creating; the subject of the predication being a specified class to which E belongs, or an indefinite part of such class." (MS [R] 842: 29-30) In more modern terms, we can define abduction (or abductive reasoning) as a mode of symbolic inference that involves the screening and selection from a domain D of the possible explanation paths to an outcome O, possibly involving any element E of D, with the selection of candidate paths for inductive testing based on plausibility, economy and potential impact. Abduction does not produce probable results, only qualified candidates (most often called hypotheses).

Redux: The Nature of Knowledge

Having discussed the three types of inferential logic, let's now turn our attention to the logical *context* for knowledge, which was a third of the emphasis in *Chapter 2*. Knowledge, after all, is not merely counting peas or tallying results but is the discovery and verification of 'facts' about the world sufficient to generate belief. A useful framework for evaluating this context goes under the ideas of *closed* or *open* worlds.

The *closed world assumption*, or *CWA*, is the presumption that what is not currently known as true is false. CWA also has a logical formalization. CWA is the most common logic applied to relational database systems and is particularly useful for transaction-type systems. In knowledge management, for which OWA is most often the best choice, we may use the closed world assumption in two situations: 1) when the knowledge base is known as complete (*e.g.*, a corporate database containing records for every employee); or 2) when the knowledge base is known as incomplete, but we must derive a 'best' definite answer from incomplete information.

The *open world assumption*, or *OWA*, is a formal logic assumption that the truth-value of a statement is independent of whether or not any single observer or agent know it. OWA directly conforms to Peirce's view of reality, knowledge, and truth. Missing values are expected and do not falsify what is there. A corollary assumption is that we will always be adding more information to the system, and the design should promote that fact. OWA is used in knowledge representation to codify the informal notion that in general no single agent or observer has complete knowledge, and therefore cannot make the *closed world assumption*. The OWA limits the kinds of inference and deductions an agent can make to those that follow from statements that are known to the agent as true (or probably true).

OWA is useful when we represent knowledge within a system as we discover it, and where we cannot guarantee that we have discovered or will discover complete information. Of course, this is the very essence of knowledge. In OWA, we may consider statements about knowledge that are not explicitly stated or inferred as unknown, rather than wrong or false.

Besides this contextual perspective, logic constructs may bring other expressive properties. Here are some of the more important ones that warrant consideration for a KR language:

- *Cardinality* — is where the number of members in a class or type is set or limited, such as `hasBiologicalParent` set to a cardinality of two;
- *Disjoint* — is where membership in one class excludes membership in another; this is a useful property in that it allows us to 'slice-and-dice' large, well-designed knowledge bases for more effective processing or analysis;
- *Domain* (property) — a statement that declares the classes or types from which to draw the subject of the assertion;
- *Function* — is any algebraic or logical expression allowable by the semantics and primitives used in the KR language where an input is related to an output;
- *Inverse* — is when a property, say, `hasParent`, can be defined as the inverse property of `hasChild`;
- *Negation* — is a unary operation that produces a value of *true* when its operand is false and a value of *false* when its operand is true;
- *Range* (property) — a statement that declares the classes or data types from which to draw the object data or types of an assertion;
- *Reflexivity* — is when every element of X is related to itself, every class is its own subclass, such as every person is a person;
- *Rules* — we may supplement the underlying logic with rules engines (if-then, exclusions, inclusions) that may add further to the specifications allowed;
- *Symmetric* — is when A relates to B exactly if it relates B with A; and
- *Transitivity* — is when item A is related to item B, and item B is related to item C, then A is also related to C; this is the critical property for establishing inheritance chains.

The use or not of these constructs both may affect how reasoners operate in a knowledge base and may add to the feature pool available to machine learners. An optimal KR language would provide all of these capabilities.

In operating KR and knowledge management systems, closed world applications can interface with the open world graph of the KR system via agreed, canonical data transfer models. Proper design can readily integrate simulation models, search engines, forecasting software, language processors, or transaction systems with the knowledge representation, enabling all parts to contribute to their strengths. Given the importance of context to knowledge representation, we devote much of *Chapter 9* to the open world topic. We discuss architectures and platform designs that enable integrating closed and open systems in *Chapter 13*.

Particulars, Generals, and Description Logics

We began our logic discussion centered on first-order logic, and its suitability to our KR needs, save for its lack of decidability. Early researchers in knowledge representation developed description logics specifically to overcome this lack, as well as other pragmatic considerations around KR.[24] Description logics are one of the underpinnings to the semantic Web. They grew out of earlier frame-based logic systems from Marvin Minsky and also semantic networks. Description logics (DLs) as a term and discipline were first defined in the 1980s by Ron Brachman, among many others.[24] DLs or fragments thereof are quite akin to FOL, but slightly less expressive, lacking negation or the unique name assumption, as examples. DLs can (usually) be made decidable, that is, able to resolve all mathematical expressions in the language, while FOL is not. Description logics firmly embrace the open world assumption, a central aspect of knowledge systems, as we continue discussing in the next chapter.

One aspect of description logics and their semantics is that they traditionally split concepts and their relationships from the different treatment of instances and their attributes and roles. This split corresponds nicely to the split between generals and particulars, respectively, that we have adopted from Peirce. In description logics, we know the concept split as the TBox (for terminological knowledge, the basis for T in TBox) and the instance split as the ABox (for assertions, the basis for A in ABox). A TBox is a conceptualization associated with a set of facts. TBox statements describe this conceptualization through a set of concepts and relationships between them. In its entirety, a TBox specifies the schema for the conceptualization; that is, an *ontology*. All generals, from a Peircean perspective, belong to the TBox.

The ABox is the complement that describes the instances (or instance *records*) and their attributes that populate that conceptualization. In these regards, extensional relationships dominate in the TBox, intensional ones in the ABox.* Though no formal or actionable difference exists between the ABox and TBox in description logics, keeping them separate is often a practical design choice.

Of course, the choice of KR logic and formalism must consider how to handle other types of relations and the whole panoply of trade-offs incurred during actual implementation, including importantly usability, toolsets and maintenance. None of these logical options prevents, in and of themselves, making inconsistent assignments or perhaps introducing cycles or other errors into our knowledge bases. Whatever logic or formalism we choose, it is essential to test for internal consistency and coherence. Keeping proven reasoners at the ready while developing is but one example of best practices when building or maintaining knowledge bases. I discuss these and related best practices in *Chapter 14*.

* One confusing aspect is that some computer science database textbooks use the term 'intension' to refer to the schema of a database, and 'extension' to refer to particular instances of a database,[25] an unfortunate use also by one of the major textbooks in description logics.[24] Peirce noted similar confusions long ago (*c.f.*, CP 2.393), and as a result tended to use the term *comprehension* over intension.

PRAGMATIC MODEL AND LANGUAGE CHOICES

The preceding discussion on logic has informed us about how to select a desirable knowledge representation language. We want a language that can model and capture intensional and extensional relations; one that potentially embraces all three kinds of inferential logic; that is decidable; one that is compatible with a design reflective of particulars and generals; and one that is open world in keeping with the nature of knowledge. We want this KR language, or languages, to accommodate Peirce's guidance, especially that related to practicality and various use and adoption criteria. In short, we seek pragmatic choices that balance the trade-off in expressivity and tractability.

Many, especially in the semantic Web community, have chosen topic maps or the Resource Description Framework (RDF) as their sole modeling basis. At the more expressive end of the spectrum, others have advocated conceptual graphs and the more powerful constructs of first-order logic. We have chosen more of a middle path: we use RDF as our data model language while using the Web Ontology Language, OWL 2, as our language for the knowledge graph, and the basis for mapping to external information sources. Both RDF and OWL 2 conform to description logics, and both are open, standardized efforts from the World Wide Web Consortium. Via these choices, we also gain access to many other standards and tools from the W3C, as the remainder of this section describes.

RDF: A Universal Solvent

RDF (Resource Description Framework) is a family of World Wide Web Consortium (W3C) specifications originally designed as a *metadata* model. In practice, RDF has become a general method for modeling information through a variety of syntax formats. In RDF, we make *statements* about *resources* in the form of *subject-predicate-object* expressions, called *triples*.

A *triple* may sound fancy, but substitute verb for *predicate* and noun for *subject* and *object*. In other words: *Dick sees Jane*; or, the *ball is round*. It may sound like a kindergarten reader, but it is how we can easily represent data and build it up into more complex vocabularies and structures. We combine multiple statements to flesh out our understanding of individual things. Since *subjects* or *objects* may act as 'nodes' to one another (the *predicate*s act as connectors or 'edges'), we may create hierarchical and relationship structures as we add statements (see *Figure 1-2*). As we aggregate these node-edge-node triple statements, a network structure emerges, known as the RDF *graph*.

The referenced 'resources' in RDF triples have unique identifiers, IRIs, that are Web-compatible and Web-scalable, such as http://mkbergman.com/me/about.rdf. These identifiers can point to precise definitions of predicates or refer to specific concepts or objects, leading to less ambiguity and clearer meaning or semantics.

We can apply RDF triples equally to *unstructured* (say, text), *semi-structured* (say, HTML documents) or *structured data* sources (say, standard databases). This flexibility makes RDF almost a 'universal solvent' for representing data structure.* By defining new types and predicates, we can create more expressive vocabularies within RDF. This expressiveness enables RDF to define controlled vocabularies with exact semantics.[26] These features make RDF a powerful data model and language for data federation and interoperability across disparate datasets.

We represent instance data simply as k*ey-value pairs (*also known as a *name-value pairs* or *attribute-value pairs*), where the *subject* is the instance (particular) itself, the *predicate* is the attribute, and the *object* is the value. We may express all or part of the data model as a collection of tuples `<attribute name,value>` where each element is a key-value pair. The key is the defined *attribute,* and the value may be a reference to another object or a literal string or value. In the base form of the RDF data model, useful in describing static things or basic facts, we keep it simple: no range or domain constraints; no existence or cardinality constraints; and no transitive, inverse or symmetrical properties. A combination of these for the same subject forms an instance *record*, part of the ABox as noted above. A *dataset* is a combination of one or more *records*, transmitted as a single unit (though we may break it into parts due to size), including simple text files.

Because of RDF's universality and open standards, a vibrant ecosystem exists of translators to alternate syntaxes, languages, and serializations, with JSON and straight text (through comma-separated value and RDF formats) being the most popular. Because of its diversity of serializations and its simple data model, it is also easy to create new converters using RDF. Generalized conversion languages such as GRDDL provide framework-specific conversions, such as for microformats. Once in a standard RDF representation, it is straightforward to incorporate new datasets or new attributes, and to aggregate disparate data sources as if they came from a single source. This universality enables meaningful composition of data from different applications regardless of format or serialization.

RDFS (RDF Schema) is the next layer in the RDF stack designed to overcome some of the baseline limitations. RDFS introduces new predicates and classes that bound these semantics. Importantly, RDFS establishes the basic constructs necessary to create new vocabularies, principally through adding the *class* and *subClass* declarations and adding *domain* and *range* to *properties* (the RDF term for *predicates*). RDFS supplies the basic *data types* used in the vocabulary, which are pre-defined ways that attribute values may be expressed, including various literals and strings (by language), URIs, Booleans, numbers, and date-times.† Many useful RDFS vocabularies exist, and it is possible to apply limited reasoning and inference support against them. We can also use this intermediate canonical form, now with a bit of added schema, to communicate queries, context selections, and labels and forms for user interfaces. The RDFS structure and label properties allow us to populate context-relevant dropdown lists and auto-complete entries in user interfaces solely from the input data and structure. This ability is generalizable using a reasonably straightforward input schema.

* See *Chapter 10.*

† See, for example, XSD (XML Schema Definition) for more information

Thus, RDF is a framework for modeling all forms of data, for describing that data through vocabularies, and for interoperating that data through shared conceptualizations and schema. We can represent, describe, combine, extend and adapt data and their organizational schema flexibly and at will using the HTTP protocol. Importantly, via existing or easily constructed converters, we can do this without the need to change what already exists. We can augment our existing relational data stores, and transfer and represent our current information as we always have. The RDF data model provides an abstract, conceptual framework for defining and using metadata and metadata vocabularies, as well as for our primary purpose of representing a message or data in a readily consumable form. In our design, RDFS is the language mostly focused on the ABox.

OWL 2: The Knowledge Graph Language

We need something more expressive and powerful for the conceptual and reasoning aspects of the TBox. Our choice, again a W3C standard, is *OWL 2,* the Web Ontology Language designed for defining and instantiating formal Web *ontologies*, or *knowledge graphs*. An OWL ontology may include descriptions of *classes*, along with their related *properties* and *instances*. A variety of OWL dialects may be employed, specialized to process more quickly for different specific needs, such as rule testing or querying. OWL 2 is the primary formalism used in KBpedia.* OWL 2 provides nearly complete capabilities from description logics and offers some tricks of its own in metamodeling. An inspection of OWL's standardized direct semantics provides further detail in these regards.[27]

Before OWL 2, the initial version of OWL was more challenging to ensure decidability. The earlier version also did not allow users to treat classes as instances depending on context. Fortunately, OWL 2 added a metamodeling technique called *punning*. When used for *ontologies*, it means to treat a thing as both a *class* and an *instance*, with use depending on context.

While we are using OWL 2 as our standard KBpedia language, we are not relying on OWL's distinction of object and datatype properties for external relations and attributes, respectively. External relations, it is true, by definition are object properties, since both subject and object are identifiable things. However, attributes, in some cases such as rating systems or specific designators, may also refer to controlled vocabularies, which can (and, under best practice, should) be object properties. So, while most attributes use datatype properties, not all may. Relations and attributes are a better cleaving since we can use relations as patterns for fact extractions and the organization of attributes gives us a cross-cutting means to understand the characteristics of things independent of the entity type. So while the splits most often conform to object properties for external relations and datatype properties for attributes, relying on this split is not dependable.† In any case, all of these assign-

* References herein to OWL refer to OWL 2 unless otherwise noted.

† For interoperability with external datasets where our logic model and vocabulary provides, we allow assignment of either object or datatype properties, which we reconcile at the schema level.

ments become valuable potential features for machine learning, in addition to the standard text structure.

OWL provides sufficient expressive richness to describe the relationships and structure of entire worldviews, or the so-called terminological (TBox) construct in description logics. Thus, we see that the complete structural spectrum of description logics can be satisfied with RDF and its schematic progeny, with a bit of an escape hatch for combining poorly defined or structural pieces using undecidable OWL fragments.

W3C: Source for Other Standards

We can use many other W3C standards in the system for graphics standards, rules, selected vocabularies, translators and validators, and the like.[28] I will mention only three of the prominent ones we use. The first capability is the RDF query language, *SPARQL*,[29] which provides querying of either the ABox or TBox or driving reports and templated data displays. Utilizing RDF's simple triple structure, SPARQL can also be used to query a dataset without knowing anything in advance about the data, which is a useful discovery mode. The second contributor is the Simplified Knowledge Organizational System (SKOS) vocabulary, which we use as a concept classification language;[30] see further discussion of SKOS below. The last notable contribution is linked data, which is a set of recommended techniques and guidelines for exposing RDF resources to the Web. It is the right technique to use if open sharing. The Linking Open Data movement that is promoting this pattern has become highly successful, with billions of useful RDF statements now available for use and consumption online.

THE KBPEDIA VOCABULARY

In *Chapter 7* we discussed the main terminological aspects to our approach to knowledge representation, grounded in Peirce's universal categories of Firstness, Secondness, and Thirdness, the topic of *Chapter 6*. We then added commentary about logical and inferencing needs, with pragmatic choices being made for the W3C languages to implement these design considerations. We are now in a position to provide a working introduction to the KBpedia vocabulary and the upper structure of its knowledge graph, the KBpedia Knowledge Ontology, or KKO. We supplement these materials with online resources and further KBpedia details; see *Appendix B.*

By design, this introduction is only a summary. KBpedia is under active use and development as of the time of this writing, and we expect details to change, perhaps in material respects. As a result, I try to keep the summary and explanations general enough to retain some longevity. Again, for the current specifications, see the online resources.*

* KBpedia and its documentation are available for free under open source licensing; for current specifications and downloads see http://kbpedia.org.

Structured on the Universal Categories

In keeping with the universal categories, we organize KBpedia under the standard RDF *root* of 'Thing'* into the three main and sole branches of *Monads* (Firstness, 1ns or 1), *Particulars* (Secondness, 2ns or 2), or *Generals* (Thirdness, also called SuperTypes, 3ns or 3). We also tend to categorize the upper structure of KBpedia, formally known as the KBpedia Knowledge Ontology, or KKO, in a triadic manner. This triad follows the senses of possible building blocks (1ns), actual things in the category (2ns), and generalizations about the category (3ns). Applicable categories in the upper structure may be prefixed with 1, 2, or 3 to keep track of these splits. We list the features available for machine learning in *Appendix C*.

The *Monads* branch (1ns) captures the qualities, constituents, characteristics, or attributes of the actual things or general realities that comprise human knowledge. We can talk about these things, but, once we do, we instantiate the quality, so that our actual statements and assertions about these things occur in the other two branches. Nonetheless, from a modeling standpoint, it is still possible to relate statements about monads to their placements in the knowledge graph, enabling some reasoning, if desired, by proxy, using this branch. The *Particulars* branch (2ns) represents all individual, real things across which knowledge may pertain. Entities and events are the two main sub-branches of the particulars, with the third sub-branch being the instantiation of monads. The *Generals* branch (3ns), the third of the three, comprises all concepts, types, and generalizations we may make about the things to which knowledge may refer, as well as the concepts and generalizations that apply to knowledge itself and how it is represented and communicated. Its three main sub-branches represent constituents of reality, relations (predications), and manifestations, including matter, life, and symbols. KBpedia's upper triadic structure is the domain of discourse for knowledge representation and its potential scope. Naturally, since it is absurd to capture all instances or all generalities related to knowledge, KBpedia is not a complete representation. Instead, it is a scaffolding of the more pivotal joints in the knowledge skeleton, which provide reference tie-ins for specific knowledge domains to expand coverage using similar construction methodologies.

By comparison, please note that most existing KR graphs or ontologies correspond to the *Generals* branch in our design, with the data or ABox corresponding to the *Particulars* branch in KBpedia and KKO. However, none of these other existing systems are triadic, and none are explicit about modeling meaning or context.

Three Main Hierarchies

We embed three hierarchical backbones within the KKO structure. One is for instances (particulars) that correspond to the ABox. One is for relations. The third is for classes, types, and generalities, corresponding to the TBox. I overview these three backbones under the instances, relations, and generals vocabulary sections below.

* 'Thing' is the same as 'resource' in RDF and is the existential starting node for an OWL knowledge graph.

The Instances Vocabulary

More than 95% of the knowledge items in KBpedia are instances, either entities or events. The constituent knowledge bases for these include Wikipedia and Wikidata, described in *Chapter 11*. All instances in KBpedia belong to one or more types, which are the subject of the *Generals* branch. However, we may use different characteristics to describe and compare instances to one another, as shown in *Table 8-2* below. These descriptions relate to how we characterize the instances, not where they occur in the general conceptual schema. These items may be discovered and inspected using the online KBpedia browser.* Hopefully, most of these items are pretty clear. You may obtain full definitions and other contextual specifications from the open source KKO artifact.

The *Monoidal Dyads* sub-branch captures the items in the *Monad* main KKO branch, previously noted, as reified as actual instances. Its triadic splits follow the general form. *Events* were discussed at length in *Chapter 7*, and capture the span from Peirce's absolute chance (*tychism*, or spontaneity) to his Thirdness of *synechism*.

- 2-Particulars
  - 1-Monadic Dyads
    - 1-Monoidal Dyad
    - 2-Essential Dyad
    - 3-Inherential Dyad
  - 2-Events
    - 1-Spontaneous
    - 2-Action
      - 1-Exertion
      - 2-Perception
      - 3-Thought
    - 3-Continuous
      - 1-Triadic Action
      - 2-Activities
      - 3-Processes
  - 3-Entities
    - 1-Single Entities
      - 1-Phenomenal
      - 2-States
        - Situations
      - 3-Continuants
        - Time
          - 1-Instants
          - 2-Intervals
          - 3-Eternal
        - Space

* See http://kbpedia.org/knowledge-graph/; links for how to use are provided on that same page.

		Points	
		Areas	
			2D-Dimensions
		Space/Regions	
			3D-Dimensions
2-Part Of Entities			
	1-Members		
	2-Parts		
	3-Functional Components		
3-Complex Entities			
	1-Collective Stuff		
	2-Mixed Stuff		
	3-Compound Entities		

*Table 8-2: Full, Upper Hierarchy of the KBpedia Particulars**

Actions are the actual instances of action and may arise from perception, exertion or thought, as previously discussed. *Entities* span from single ones, according to the universal categories, to parts of entities and then combinations of entities, again in correspondence with the categories. *Complex entities* may span from simple collections to mixtures to compound ones, the latter best exemplified by the constituent entities comprising the whole universe.

To my knowledge, no existing knowledge graph or ontology other than KBpedia provides a similar classification scheme for the nature of instances, likely because none of the other systems are modeled using Peircean perspectives.

The Relations Vocabulary

I provided a fairly detailed introduction to the *Relations* vocabulary in *Chapter 7*. We model these relations as abstract possibilities under the relational monads (2ns and 3ns) in the *Monads* main branch. We model these relations as concepts used in knowledge representation according to the *Predications* branch of 2ns in *Table 10-2*. Note KKO represents the concepts of these relations, in addition to the relational expressions themselves. In KKO, we provide the specific relations as object or datatype properties (or both), depending. We include the separate listing of relations as classes so that we may talk about and reason over them as *concepts*. Using them in actual triples requires the properties.

Since we already introduced the top-level of the relations in *Chapter 7*, let's move on to the next two levels. The next *Table 8-3* shows the second level of the relations hierarchy, with the following *Table 8-4* showing the third-level. I will highlight some aspects of these tables where they may not be entirely evident, according to the 1ns, 2ns, and 3ns relation sub-branches.

* Various downloads of KKO and KBpedia may be obtained from https://github.com/Cognonto/kko. To view the KKO artifact, you will need an ontology editor, such as the open source Protégé ontology development environment.

Attributes Relations (1ns)

Attributes are the intensional characteristics of an object, event, entity, type (when viewed as an instance), or concept. We split attributes into three categories. The *intrinsic* relations are innate characteristics or essences of single entities or events (particulars). Example concepts include oneness, qualities, feelings, inherent, negation, is, has, intensional, naturalness, internal, innateness. *Qualities*, one intrinsic sub-branch, are an internal characteristic or aspect of an object; collectively these define intensionally what types to which the object belongs, though that relationship is not intrinsic. *Elementals* are a contributing part of or integral input or aspect that adds to the understanding of the subject. *Configurations*, or forms or arrangements, are of the nature or perceivable of the subject. The *adjunctual* are occurrences that may occur to single entities or events (particulars) that help characterize it. Example concepts include birth, death, marriage, events for the individual, accidents, surprises, happenings, extrinsic, adjunctual. Though Peirce used 'accidental' much, he applied it in most cases to 'accidental actuals'; thus, 'adjunct' better captures potentiality. Within adjunctuals we have *quantities*, characteristics of a subject that we express as a numerical quantity; *eventuals*, chance, accidental or planned occurrences that directly involve a subject; or *extrinsics*, which are external events or circumstances that directly involve the subject or help define the nature or reality of it.

The *contextual* relations are circumstances or placements of single entities or events (particulars) that help characterize it. Example concepts include space, time, continuity, and classificatory. These relations include anything that has gradation over space and time, including ideas and concepts that also shade. The three sub-branches of the contextual relations are: *situants* (1ns), which are attributes or characteristics that help situate, or place the subject in a locational or time context; *ratings (2ns)*, which are an assigned value or characterization that orders the subject in relation to other subjects; or *classifications (3ns)*, which are characterizations of the subject in regard to multi-factor typing, coding or value in relation to a given attribute or set of attributes.

External Relations (2ns)

External relations are assertions between an object, event, entity, type, or concept and another particular or general. *External relations* also have three sub-categories, with the first (1ns) being *direct*, which are simple relationships (no intermediaries) between two different objects considered as instances. Example concepts include *is a*, simple without parts, part of, members in types or classes, or genealogical roles (parent, child, brother). Direct relations, in turn, have three sub-branches, including: *equivalences*, a simple relationship that asserts equality or sameness; *parts*, a simple relationship where the object is a part or component of the subject; or *descendants*, which are a simple relationship where the object has a genealogical, subsumption, or supersumption relationship to the subject.

The *Copulative* relations are the 2ns sub-branch of the *External Relations*. They convey combination, membership, quantity, action, or joins.[31] The three sub-branches include *typings*, all of the *is-a* relations to types; *actions*; and *conjoins*, relations that involve the joining of a subject to an object via an intermediate object. We should note that the two sub-branches of *external relations* to this point, the *direct* and *copulative*, represent the *simple* relations according to Peirce's logic of relatives.[32]

The last sub-branch in Thirdness of the *External Relations* is the *Mediative* relations, which are the true, triadic external relations, such as 'A gives B to C.' These are the relationships of relevance, meaning, explanation, or cognition. Sub-branches of the mediative relations are *comparisons*; *performances*, which are relations of quantity or rank for how a subject performed in relation to an object; or *cognitives*, which relate to thinking, knowing or representing. While we might consider thoughts as something that occurs internally, thoughts are not innate and are internal representations of the external world.

Representation Relations (3ns)

The Thirdness branch of relations is the *Representations*, which are signs (1905, CP 8.191) and the means by which we point to, draw or direct attention to, or designate, denote or describe a particular object, entity, event, type or general. The first Representation sub-branch is the *denotatives*, icons or symbols that name or describe the subject. Its three sub-branches are: *media*, iconic images or sounds that invoke the identification with a given object or representation; *labels*, symbolic text strings that help name or draw attention to a particular object; or *descriptions*, text strings that may be longer than labels and provide additional or contextual information or specify attributes about the object, beyond drawing attention.

The second branch of the *Representations* is the *indexes*, indirect references or pointers that help draw attention to the subject. *Indexes* are references or attention-directors to a subject.[33] The three sub-branches of *indexes* are: *pointers*, physical or symbolic indicators of a given thing and which draw attention to it; *identifiers*, such as URIs, which are generally (unique) symbols or strings that provide a key to a given subject, often within some conventional scheme for generating and recognizing the token assigned; and *codes*, an assigned symbolic token or string that groups the object with similar items.

The last branch (3ns) of the *Representations* is the *associatives*, contextual assertions of proximity, affiliation or adjacency of the subject to any contiguity.[34] The three sub-branches are: *lists*, either ordered or unordered aggregations of objects similar to one another with respect to given characters or types; *relateds* (*see also*), which are indicators of some nature to other objects similar or related to it; or *augments*, which are an external indicator that leads to still further explanations.

Current practice rarely incorporates any of these vocabulary aspects — discussed in the sections above on *monads*, *particulars*, and *relations* — in knowledge graphs and ontologies. The Peirce-inspired design of these first and second branches of KKO demonstrate a logical, recursive approach to organizing knowledge domains. These aspects provide a deeper, more abstract pool of features of possible use to machine learners. Reasoning tasks should also see a jump-step up in capabilities.

The Generals (KR Domain) Vocabulary

The core of KBpedia, as it is for most current knowledge graphs, is the TBox, or the conceptual schema for the domain. This KKO branch is populated entirely with generals and is where most current reasoning with knowledge graphs occurs. This central schema is also the point at which it is best to link external knowledge bases into the system.* Because of this mapping role, we term the nodes in the *Generals* branch as *reference concepts* (or *RefConcepts* or *RCs*). All RCs are OWL *classes*. More than 95% of the 55,000 current *base concepts* in KBpedia are RCs. These RCs provide a rich pool of tie-in points for enabling integration with external sources.

The RCs are organized into natural hierarchies of related kinds or *types*, what we term *typologies*.† Typologies are multi-instance hierarchies; each one has a top-level node called a *SuperType*. The distribution of typologies in KBpedia covers the scope of substantive human knowledge, and all of the SuperTypes, by design, are part of the upper KKO knowledge graph. Also, we design the typologies as *disjoint* (non-overlapping) with one another where possible, which promotes efficiency in reasoning and other analyses. Typologies are explained further in *Chapter 10*.

We provide a complete view of the upper *Generals* branch in *Table 10-1*, in the chapter on *typologies*. The structure of the *Generals* branch follows our understanding of Peirce's universal categories. Note the structure enables us to organize and reason over predicates and attributes, as well as the more standard classes of things that encompass the knowledge domain. Further, via ties to the other two main KBpedia branches, *Monads* and *Particulars*, we can also significantly expand the abstract characterization and reasoning of all things within that domain.

Other Vocabulary Considerations

Before we close out discussion of the KBpedia vocabulary, we need to touch upon two further considerations: the vocabulary terms provided by W3C standards and the vocabulary for mapping external sources to KBpedia. As the following *Table 8-5* shows, we rely much on the SKOS vocabulary in KBpedia for various annotation labels and some conceptual relationships. We use RDFS for SKOS, property range and domain declarations, and property hierarchies. OWL is used to declare classes and to split our properties into annotations, object properties, and datatype properties.

SKOS, or the Simple Knowledge Organization System,[30] is a formal language and schema designed to represent such structured information domains as thesauri, classification schemes, taxonomies, subject-heading systems, controlled

* While instance mappings are possible, it is more effective to define relationships at the class level, since member instances can then be inherited without direct assignment.

† These are a critical design component of our approach, which we discuss at length in *Chapter 11*.

vocabularies, or others; in short, most of the 'loosely defined' ontology approaches discussed herein. It is a W3C initiative more fully defined in its SKOS Core Guide. As an RDF Schema, SKOS adds some language and defined relationships to the RDF baseline. SKOS also has a rich set of annotation and labeling properties to enhance the human readability of schema developed in it.

RDFS	rdfs:domain rdfs:range rdfs:subClassOf rdfs:subPropertyOf
OWL	owl:AnnotationProperty owl:Class owl:DatatypeProperty owl:disjointWith owl:equivalentClass
SKOS-Preferred	skos:altLabel skos:broaderTransitive skos:definition skos:hiddenLabel skos:narrowerTransitive skos:prefLabel
SKOS-Optional	skos:broader skos:changeNote skos:editorialNote skos:example skos:historyNote skos:narrower skos:note skos:related skos:scopeNote

Table 8-5: External Mapping and Annotation Properties

As noted, the *Generals* branch is the target for mapping to external sources. The design approach is to define the classes in KBpedia broadly and to consider external mappings of the `subClassOf` nature. What this means is that the parental concept in KBpedia tends to subsume the concepts in the contributing external sources, and to, therefore, inherit the instances brought in by external classes.* However, not all mappings represent class-to-class relationships. Further, some mappings may be more of the nature of intersections or partial overlaps, rather than complete inheritance. As a result, KBpedia has adopted multiple mapping predicates, some approximate, as shown in *Table 8-6*:

* Due to OWL 2 and punning (*c.f.*, *Chapter 9*), depending on context, we can talk about the classes in the *Generals* branch as instances and characterize them, while they can still act as classes for mapping and logical inheritance purposes.

correspondsTo	The property *correspondsTo* is used to assert a close correspondence between an external class, named entity, individual or instance with a Reference Concept class. correspondsTo relates the external class, named entity, individual or instance to the class by both its subject matter and intended scope. This predicate should be used where the correspondence between the two entities is felt to be nearly equivalent to a *sameAs* assertion and is reflexive, but without the full entailments of intensional class memberships. In these cases, both entities are understood to have the same type and intended scope, but without asserting a full class-level or *sameAs* individual relationship. This predicate is for aligning two different ontologies or knowledge bases based on node-level correspondences, but without entailing the actual ontological relationships and structure of the object source. For example, the *correspondsTo* predicate may be used to assert close correspondence between Reference Concepts and Wikipedia categories or pages, yet without entailing the actual Wikipedia category structure. This property asserts a different and stronger relationship than *isAbout*.
isAbout	The property *isAbout* is used to assert the relation between an external named entity, individual or instance with a Reference Concept class. *isAbout* relates the external named entity, individual or instance to the class by its subject matter. The relation acknowledges that the scope of the class cannot be determined solely by the aggregation or extent of its associated individual entity members and that the nature of the Reference Concept class may not alone bound or define the individual entity. This property is therefore used to create a topical assertion between an individual and a Reference Concept.
isRelatedTo	Check the definition of *isAbout* for the definition of this property; *isRelatedTo* is the inverse property of *isAbout*.
relatesToXXX	The various properties designated by *relatesToXXX* are used to assert a relationship between an external instance (object) and a particular (*XXX*) SuperType. There may be as many *relatesToXXX* properties as there are numbers of SuperTypes. The assertion of this property does not entail class membership with the asserted SuperType. Rather, the assertion may be based on particular attributes or characteristics of the object at hand. For example, a British person might have a *relatesToXXX* asserted relation to the SuperType of the geopolitical entity of Britain, though the actual thing at hand (person) is a member of the Person class SuperType. This predicate is used for filtering or clustering, often within user interfaces. Multiple *relatesToXXX* assertions may be made for the same instance.
isLike	The property *isLike* is used to assert an associative link between similar individuals who may or may not be identical, but are believed to be so. This property is not a general expression of similarity, but rather the likely but uncertain same identity of the two resources. This property is an alternative to *sameAs* where there is not a certainty of sameness, and when it is desirable to assert a degree of overlap of sameness via the *hasMapping* reification predicate. This property can and should be changed if the certainty of the sameness of identity is subsequently determined. *isLike* has the semantics of likely identity, but where there is some uncertainty that the two resources indeed refer to the same individual with the same identity. Such uncertainty can arise when, for example, we use common names for different individuals (*e.g.,* John Smith). It is appropriate to use this property when there is strong belief the two resources refer to the same individual with the same identity, but that association cannot be made at present with full certitude.

hasMapping	The *hasMapping* property is used to reify *isAbout*, *isRelatedTo* or an *isLike* property assertion with a statement as to its degree of mapping or relationship between subject and object. The *hasMapping* property may be expressed as a mapping percentage value, some quantitative metric value, or a qualitative descriptor characterizing the linkage degree or overlap between the two classes, predicates, individuals or datatypes. This value might be calculated from some external utility, may be free form, or may be based on some defined listing of mapping values expressed as literals.
hasCharacteristic	The property *hasCharacteristic* is used to assert the relation between a Reference Concept, or any other classes, and external properties that may be used in external ontologies to characterize, describe or provide attributes for data records associated with that concept or that class. It is via this property or its inverse, *isCharacteristicOf,* that external data characterizations may be incorporated and modeled within a domain ontology based on the KBpedia vocabulary.
isCharacteristicOf	The property *isCharacteristicOf* is used to assert the relation between a property and a Reference Concept (or its punned individual), or any other classes, to which it applies. Such properties may be used in external ontologies to characterize, describe, or provide attributes for data records associated with that concept or that class. It is via this property or its inverse, *hasCharacteristic*, that external data characterizations may be incorporated and modeled within a domain ontology.

Table 8-6: Mapping and Alignment Relations

We cover the general topic of mapping in some detail in *Chapter 13*.

Chapter Notes

1. Some material in this chapter was drawn from the author's prior articles at the *AI3:::Adaptive Information* blog: "Thinking 'Inside the Box' with Description Logics" (Nov 2008); "'Structs': Naïve Data Formats and the ABox" (Jan 2009); "Advantages and Myths of RDF" (Apr 2009); "Uses and Control of Inferencing in Knowledge Graphs" (Mar 2017); "KBpedia Relations, Part V: The Updated KBpedia Grammar" (Jul 2017); "Why I Study CS Peirce" (Aug 2017).
2. Davis, R., Shrobe, H., and Szolovits, P., "What Is a Knowledge Representation?," *AI Magazine*, vol. 14, 1993, p. 17.
3. Bergman, M. K., "The Rationale for Semantic Technologies," *AI3:::Adaptive Information*, Jul. 2012.
4. Huber, F., "Formal Representations of Belief," *Stanford Encyclopedia of Philosophy*, Oct. 2008.
5. Roche, W., "Coherence and Probability: A Probabilistic Account of Coherence," *Coherence: Insights from Philosophy, Jurisprudence and Artificial Intelligence*, Springer, 2013, pp. 59–91.
6. Douven, I., and Meijs, W., "Measuring Coherence," *Synthese*, vol. 156, May 2007, pp. 405–425.
7. Brachman, R. J., and Levesque, H. J., *Knowledge Representation and Reasoning*, Morgan Kaufmann, 2004.
8. Levesque, H. J., "Making Believers Out of Computers," *Artificial Intelligence*, vol. 30, 1986, pp. 81–108.
9. Levesque, H. J., and Brachman, R. J., "Expressiveness and Tractability in Knowledge Representation and Reasoning," *Computational Intelligence*, vol. 3, 1987, pp. 78–93.
10. Zhou, Y., "A Set Theoretic Approach for Knowledge Representation: the Representation Part," *arXiv*, vol. 1603, Mar. 2016.
11. Zeman, J. J., "The Graphical Logic of CS Peirce," Ph.D., University of Chicago, 1964.
12. Considerable literature deals with this area (search repositories with "set theory" and "knowledge representation"), but set theory is not as developed as FOL and does not have the relationship to Peirce.

13. The *intension* of a class comprises what characteristics its instances have. Intension is most akin to the *attributes* or characteristics of the instances in a set defining its class membership.
14. The *extension* of a class comprises the enumeration of its instances, consisting of the things to which it subsumes. Most hierarchical relationships in contemporary knowledge bases are explicit assignments to class or type, by definition extensional. Extensional enumeration alone is not feasible for large knowledge bases.
15. "Knowledge Representation and Reasoning," *Wikipedia*, Oct. 2017.
16. Sowa, J. F., "Conceptual Graph Summary" Available: http://www.jfsowa.com/cg/cgif.htm.
17. Patterson, E., "Knowledge Representation in Bicategories of Relations," *arXiv*, vol. 1706, Jun. 2017.
18. Sdrolia, C., "Signifying Nature: Semeiosis as the Foundation of Post-Critical Cosmology in Charles S. Peirce," Goldsmiths, University of London, 2014.
19. Staat, W., "On Abduction, Deduction, Induction and the Categories," *Transactions of the Charles S. Peirce Society*, vol. 29, 1993, pp. 225–237.
20. Muggleton, S., "Inductive Logic Programming," *New Generation Computing*, vol. 8, 1991, pp. 295–318.
21. d'Amato, C., Fanizzi, N., Fazzinga, B., Gottlob, G., and Lukasiewicz, T., "Combining Semantic Web Search with the Power of Inductive Reasoning," *International Conference on Scalable Uncertainty Management*, Springer, 2010, pp. 137–150.
22. De Raedt, L., *Logical and Relational Learning*, Springer Science & Business Media, 2008.
23. Völker, J., and Niepert, M., "Statistical Schema Induction," *The Semantic Web: Research and Applications*, 2011, pp. 124–138.
24. Baader, F., Calvanese, D., McGuiness, D., Nardi, D., and Patel-Schneider, P., *The Description Logic Handbook: Theory, Implementation and Applications*, Cambridge University Press, 2003.
25. See https://en.wikipedia.org/wiki/Extension_(semantics), retrieved December 6, 2017.
26. *RDF Semantics*, World Wide Web Consortium, 2004.
27. Motik, B., Patel-Schneider, P. F., and Grau, B. C., *OWL 2 Web Ontology Language Direct Semantics (Second Edition)*, World Wide Web Consortium, 2012.
28. Consult the *World Wide Web Consortium*'s W3C Web site at https://www.w3.org/.
29. Feigenbaum, L., and Prud'hommeaux, E., "SPARQL by Example," *Cambridge Semantics*, May 2013.
30. *SKOS Simple Knowledge Organization System Reference: W3C Recommendation*, World Wide Web Consortium, 2009.
31. To gain a feel for this relation, see (in English), https://en.wikipedia.org/wiki/List_of_English_copulae.
32. Peirce, C. S., "The Logic of Relatives," *The Monist*, 1897, pp. 161–217.
33. Here is a supporting quote from Peirce on the notion of *index*: "Indices may be distinguished from other signs, or representations, by three characteristic marks: first, that they have no significant resemblance to their objects; second, that they refer to individuals, single units, single collections of units, or single continua; third, that they direct the attention to their objects by blind compulsion. But it would be difficult if not impossible, to instance an absolutely pure index, or to find any sign absolutely devoid of the indexical quality. Psychologically, the action of indices depends upon association by contiguity, and not upon association by resemblance or upon intellectual operations" (1901, CP 2.306).
34. "Association is the only force which exists within the intellect, and whatever power of controlling the thoughts there may be can be exercised only by utilizing these forces; indeed, the power, and even the wish, to control ourselves can come about only by the action of the same principles. Still, the force of association in its native strength and wildness is seen best in persons whose understandings are so little developed that they can hardly be said to reason at all. Believing one thing puts it into their heads to believe in another thing; but they know not how they come by their beliefs, and can exercise no control over the inferential process. These unconscious and uncontrolled reasonings hardly merit that name; although they are very often truer than if they were regulated by an imperfect logic, showing in this the usual superiority of instinct over reason, and of practice over theory." (1886, CP 7.453).